UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA DE TELECOMUNICACIÓN

PROYECTO FINAL DE CARRERA

Estudio de un IDS *Open Source* frente a herramientas de análisis y explotación de vulnerabilidades

AUTOR: Hugo Gascón Polanco

TUTOR: Agustín Orfila Díaz-Pabón

TÌTULO: *Estudio de un IDS Open Source frente a herramientas de análisis y explotación de vulnerabilidades.*

AUTOR: *Hugo Gascón Polanco*

TUTOR: *Agustín Orfila Díaz-Pabón*

La defensa del presente Proyecto Fin de Carrera se realizó el día 11 de Febrero de 2010; siendo calificada por el siguiente tribunal:

PRESIDENTE: *Arturo Ribagorda*

SECRETARIO: *Sergio Pastrana*

VOCAL: *Javier Fernández*

Habiendo obtenido la siguiente calificación:

CALIFICACIÓN: *Matrícula de Honor*

Presidente **Secretario** **Vocal**

A mi familia.

The user's going to pick dancing pigs over security every time.

Bruce Schneier

Resumen

El presente documento describe el procedimiento seguido para la realización del Proyecto de Fin de Carrera del alumno Hugo Gascón Polanco y supone el último requisito académico para la obtención del título de Ingeniero Superior de Telecomunicación en la universidad Carlos III de Madrid.

Ante el vertiginoso ritmo de cambio inherente al mundo de la seguridad de los sistemas de información y la necesidad constante de actualización de los sistemas implicados en la misma, surge la motivación original de este proyecto: analizar la respuesta desde un punto de vista tanto teórico como práctico de Snort, un sistema de detección de intrusiones en red *open source* de amplia difusión, frente a herramientas de explotación y detección de vulnerabilidades.

El sistema de detección de intrusiones (IDS) es una herramienta básica a la hora de garantizar la integridad y disponibilidad de los sistemas en redes IP, de ahí, el interés en analizar la adecuación de la respuesta ante diversas herramientas que se encuentran en competición directa con el mismo y que permiten identificar vulnerabilidades en auditorías de sistemas o en muchos casos, realizar ataques para aprovechar vulnerabilidades accesibles de forma remota.

Snort es un IDS desarrollado por Sourcefire [5] bajo licencia GNU y su funcionamiento está basado en la identificación de ataques por reglas, permitiendo inspeccionar tanto protocolos como anomalías en el tráfico de red. Ha sido descargado por millones de usuarios y existen más de 225.000 registrados, por lo que podríamos decir que Snort es un estándar de facto en cuanto a tecnología de detección de intrusiones.

Aunque existen distintos enfoques y métricas a la hora de determinar la eficacia de este tipo de sistema, la rapidez de los procesos de actualización y el número de ataques que pueden ser detectados son indicadores significativos de su efectividad. El primer paso de este proyecto

ha sido determinar, mediante un análisis comparativo, las ventanas de vulnerabilidad de Snort frente a la aparición de nuevos fallos y respecto a la actualización de Nessus, una herramienta de detección de vulnerabilidades que se utiliza de forma habitual para comprobar la respuesta del sistema de detección de intrusiones. Para ello, se han analizado datos relativos a los ataques a vulnerabilidades identificados por Snort y las fechas de aparición de las mismas. Se han generado gráficas y medidas estadísticas de los retardos de actualización que han permitido verificar la existencia de dichos retardos y ofrecer valores comparativos de los mismos. Por último, se ha demostrado cómo los procesos de actualización de las aplicaciones vulnerables dan lugar a ventanas de vulnerabilidad menores que aquellas asociadas a la actualización de las aplicaciones destinadas a protegerlas, es decir, los sistema de detección de intrusiones y de análisis de vulnerabilidades.

Una vez conocidos, de forma teórica y estadística, los retardos presentes en la actualización del sistema detector de intrusiones, y tras la configuración de un entorno de pruebas adecuado, con los elementos necesarios para simular procesos de ataque y detección (atacante, objetivo y detector de intrusiones), se han realizado diversas pruebas de concepto para analizar de forma experimental la respuesta de Snort ante herramientas de detección y explotación de vulnerabilidades. Las herramientas seleccionadas son aquellas que son habitualmente utilizadas en las distintas etapas de un ataque: Nmap, un *software* de análisis de puertos para la identificación remota de los sistemas o *fingerprinting*, Nessus, para la identificación de las vulnerabilidades existentes en los mismos, y el entorno de desarrollo y ejecución remota de *exploits* Metasploit, que permite la explotación de las vulnerabilidades con el objetivo de tomar el control de los sistemas atacados.

Para analizar la respuesta de Snort ante intentos de detectar los servicios disponibles en un sistema de red, se han realizado distintas pruebas con Nmap. Se han utilizado diferentes configuraciones de este analizador de puertos y se han aplicado técnicas de ocultación y evasión del análisis para comprobar la fiabilidad de la respuesta de Snort. Se ha comprobado cómo las estrategias de detección de Snort se encuentran adecuadas a los mecanismos de análisis de Nmap, obteniéndose resultados positivos en cuanto a identificación y detección de los ataques de reconocimiento.

En segundo lugar se han realizado diferentes pruebas de concepto en las que se ha intentado verificar la presencia de servicios vulnerables en un sistema configurado para este fin y se ha comprobado la respuesta de Snort ante el sistema de detección de vulnerabilidades. Para

ello se ha utilizado el escáner Nessus, habitual en tareas de auditoría y que ha permitido poner de manifiesto algunas debilidades en cuanto a su efectividad a la hora de verificar el funcionamiento de un IDS, como la dificultad para identificar de forma unívoca la respuesta del sistema detector de intrusiones ante un ataque o la generación de ruido con envío de tramas ajenas a la vulnerabilidad cuya respuesta se busca analizar.

Por último, se ha analizado la respuesta de Snort ante el ataque directo realizado con *exploits*, fragmentos de código que permiten aprovechar una vulnerabilidad en un sistema permitiendo el control del mismo por el atacante o provocando un fallo en el servicio. El entorno de ejecución de código malicioso Metasploit, ha permitido realizar una serie de ataques estandarizados contra un sistema *linux* preparado como objetivo. Con la realización de esta serie de ataques se ha comprobado la respuesta de Snort ante la presencia de *exploits* individuales y la detección realizada por su motor de reglas, diseñado específicamente para detectar este tipo de código. Se muestra como a pesar de la dificultad para realizar un ataque exitoso sobre un sistema vulnerable, el sistema detector de intrusiones permite identificar de forma limitada aunque satisfactoria tales intentos de explotación.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1. La seguridad de las redes y los sistemas de detección de intrusiones

1.1.1. Una visión de la seguridad de los sistemas de información

En los últimos años, con la democratización del acceso a Internet, de los ordenadores personales, los teléfonos móviles de última generación, *netbooks* y demás dispositivos de comunicaciones, la tasa de uso de la red continúa incrementándose. Es, sin duda, un fenómeno imparable que está cambiando la vida de las personas de forma gradual, modificando aspectos cotidianos como el estudio, las actividades de ocio y recreo, las formas de comunicarse con otras personas o incluso las compras a través de la red.

Además de la gente común, las estructuras de las empresas y los negocios también participan de una profunda transformación en todos sus procesos gracias a Internet y los avances en las redes de comunicaciones. De este modo y con el objetivo de conseguir mejorar y alcanzar una mayor eficiencia en su actividad, muchas empresas y organizaciones gubernamentales han desarrollado y continúan desarrollando numerosas aplicaciones y servicios cuyo funcionamiento está ligado a Internet.

Sin embargo y a pesar de que Internet proporciona un sinfín de ventajas, trae consigo el grave problema de la seguridad de la información, que puede ser robada o modificada desde cualquier parte del mundo gracias a las mismas redes que aportan los incontables avances con los que nos maravillamos día a día. Un ejemplo común es el de servidores que son atacados e

inutilizados, y cuyos datos son robados y, en el peor de los casos, destruidos. Cuando algo así ocurre, se producen las consiguientes pérdidas económicas para las víctimas de tales ataques. En el año 2000, por ejemplo, American Yahoo fue víctima de un ataque de denegación de servicio distribuido (*DDos*). Los servidores estuvieron inaccesibles durante, aproximadamente, 3 horas, un millón de usuarios sufrieron la indisponibilidad del servicio y las pérdidas para la empresa fueron demasiado grandes para siquiera poder calcularlas. Otras empresas famosas en Internet como CNN, eBay, Amazon.com, Buy.com, etc. también sufrieron, durante la pasada década, ataques similares.

Gracias al revolucionario sistema de acceso a la información que supone Internet, es relativamente sencillo para la persona suficientemente motivada, acceder a diferentes técnicas y métodos de ataque que poder utilizar contra los servicios disponibles en la red. Los delincuentes requieren progresivamente menos conocimiento especializado para acceder a herramientas que les permitan atacar diversos sistemas con intereses cada vez más alejados de causas románticas y más cercanos a los puramente económicos o incluso en algunos casos, geopolíticos. Como consecuencia de ello, la tasa de ataques a infraestructuras de red continúa incrementándose año a año según las estadísticas de organismos oficiales como el *American Computer Emergency Response Team/Coordination Center (CERT/CC)*. Durante los últimos años, los ataques a través de internet han llegado a convertirse en una nueva arma militar. Algún ejemplo lo encontramos en las fuerzas militares de China que ya se especializan con la intención de atacar a los grupos de misión aérea norteamericanos con el objetivo de hacerles perder capacidad de combate en el aire en caso de un posible conflicto y todo ello, a través de Internet. Otro ejemplo reciente son los ataques sufridos por distintas páginas comerciales y gubernamentales en Korea del Sur y Estados Unidos con origen en Korea del Norte en julio de 2009. Se vieron afectados los servidores web del departamento del tesoro, el servicio secreto, la comisión federal de comercio y el departamento de transporte americanos. Los expertos consultados confirmaron que no se trataba de una acción individual, sino de un ataque planeado y ejecutado por una organización de tamaño considerable e incluso posiblemente apoyada a nivel de estado [17]. En cualquier caso, tal información revela la urgente necesidad de identificar y mitigar los ataques a las infraestructuras de red a través de Internet.

Las empresas utilizan, de forma habitual, el cortafuegos como primera linea de defensa para garantizar la seguridad en Internet, pero la función principal de este dispositivo es supervisar los comportamientos de acceso desde la red externa y posee capacidades limitadas en cuanto

a detección de ataques específicos.

El sistema de detección de intrusión o IDS de sus siglas en inglés *Intrusion Detection System*, se define como un dispositivo de supervisión y alarma, ya sea software o hardware diseñado para detectar intentos no autorizados de acceso, manipulación o interrupción del servicio prestado por sistemas informáticos accesibles a través de la red. El IDS permite observar y analizar la posibilidad de que se esté produciendo un ataque en tiempo real. Posee la capacidad de generar una alarma antes de que los ataques puedan ocasionar riesgo para la infraestructura de red y de ejecutar medidas oportunas para evitar que se produzcan mayores pérdidas [24].

1.1.2. El papel del IDS

Un IDS puede considerarse el equivalente *high-tech* de una alarma antirrobo, configurada, eso sí, para monitorizar pasarelas de información, actividades hostiles e intrusos conocidos. Es una herramienta especializada, capacitada para capturar e interpretar el tráfico de red o las actividades de las estaciones de trabajo. Toda la información analizada puede incluir desde paquetes capturados directamente desde la red, archivos de registro de los routers (logs), de cortafuegos y servidores, de sistemas locales y peticiones de servicio, datos sobre flujos de red y muchas otras posibilidades. En la mayoría de los casos un IDS posee una base de datos de firmas de ataques conocidos que pueden ser comparadas con patrones de actividad, tráfico o comportamiento para generar una alerta o diferentes tipos de acciones automatizadas cuando se produce una identificación positiva de un ataque, desde desconectar ese flujo de la red, a lanzar *back-traces* que permitan identificar al atacante y obtener evidencias de sus actividades. Por analogía, un IDS realiza en la red lo que un software antivirus realiza sobre los ficheros que acceden o que son copiados a un sistema: inspecciona el contenido del tráfico de red para buscar y anular posibles ataques, igual que el software antivirus inspecciona el contenido de nuevos ficheros, adjuntos de e-mail o contenido activo en una web para identificar patrones que concuerden con *malware* conocido o acciones maliciosas.

Como los cortafuegos, los IDS pueden estar basados en software o pueden combinar hardware y software en forma de dispositivos independientes preconfigurados y preinstalados. El software de un IDS puede funcionar sobre el mismo dispositivo en el que está funcionando un servidor, un cortafuegos, un proxy o cualquier otro servicio perimetral, aunque es más común

encontrar configuraciones donde el sensor IDS se encuentra separado de los demás dispositivos de la red. En cualquier caso y a pesar de una configuración independiente, el IDS monitorizará especialmente este tipo de dispositivos, que aunque se encuentran habitualmente en el perímetro de la red, no impiden al IDS detectar y tratar con ataques originados tanto desde el interior de la red, como desde un segmento externo. En numerosas ocasiones, son utilizados también para detectar violaciones de la política de seguridad especifica de una empresa o entorno corporativo de red.

Existen varios tipos de IDS atendiendo al tipo de actividades, tráfico, transacciones o sistemas que monitorizan. Los IDS que supervisan los enlaces de la red y redes troncales se denominan IDS de red o NIDS (*Network Intrusion Detection System*), mientras que los que operan en hosts y defienden y monitorizan el sistema operativo y los sistemas de ficheros buscando signos de intrusión, se suelen llamar *host-based IDSs*. En ocasiones, se configuran grupos de IDSs funcionando como sensores remotos e informando a una estación central y se conocen como IDSs distribuidos. Un *gateway IDS* es un NIDS desplegado en la puerta de enlace entre dos redes y que monitoriza el tráfico intercambiado entre ellas. Los IDSs enfocados a analizar el flujo de tráfico específico de una aplicación en relación con la lógica de la misma, además de los protocolos sobre los que se asienta su funcionamiento se conocen como *application IDSs*. En la práctica, la mayoría de los entornos comerciales utilizan una combinación de IDS de red-host y/o aplicación para observar lo que está ocurriendo en la red a la vez que se supervisan hosts relevantes y sus aplicaciones.

Los IDS también pueden ser clasificados según el enfoque que aplican al análisis de eventos. Algunos sistemas utilizan una técnica llamada "detección por firma"*(signature detection)*, similar a la que utilizan los antivirus para identificar código malicioso y bloquear los ficheros, programas o contenido web malicioso para evitar su entrada al ordenador. La diferencia radica en que el IDS utiliza una base de datos de tráfico o patrones de actividad relacionados con ataques conocidos llamados firmas. De hecho, la detección por firmas o reglas, es el más común de los métodos utilizados en los IDSs comerciales hoy día. Otro enfoque es el conocido como "detección de anomalías". En este caso, el IDS utiliza reglas o conceptos predefinidos sobre lo que puede considerarse actividad *normal* y *anormal* del sistema (*heurística)*. El objetivo es distinguir correctamente las anomalías del funcionamiento correcto y monitorizar, informar o bloquear las mismas cuando se producen. Algunos detectores de anomalías implementan perfiles de usuario, que están basados en líneas esenciales de lo que comprende la

actividad normal de un tipo de usuario específico y pueden ser construidas mediante muestreo estadístico, reglas o redes neuronales, etc.

1.1.3. El IDS de red o NIDS

El NIDS debe su nombre al hecho de que monitoriza completamente la red desde la perspectiva del lugar donde ha sido desplegado. Propiamente dicho, supervisa su segmento de red. Habitualmente, la interfaz de red de un ordenador (NIC) funciona en modo no promiscuo, es decir, sólo los paquetes destinados a la dirección de control de acceso al medio (MAC) de esa interfaz o los paquetes de difusión son procesados por capas superiores de la pila de protocolos. El NIDS deber operar en modo promiscuo para monitorizar el tráfico de la red no destinado a su propia dirección MAC. De este modo, el NIDS puede escuchar todas las comunicaciones del segmento de red sin ser detectado siempre que esté conectado a un puerto del switch local y éste haya sido configurado para replicar todo el tráfico hacia ese puerto.

La figura 1.1 representa, a modo de ejemplo, una red donde se han desplegado dos IDSs de red en segmentos estratégicos de la misma, desde donde es posible supervisar el tráfico de todos los demás elementos de la red. Esta configuración representa una topología estándar de una red perimetral de seguridad en la que las subredes en las que se encuentran los servidores públicos están protegidas por el detector de intrusiones. Cuando un servidor público se ve comprometido en una red monitorizada, el servidor puede convertirse en una pasarela para el lanzamiento de nuevos exploits, por lo que una supervision cuidadosa para evitar más daños se hace imprescindible.

Las estaciones de trabajo de la subred interna están protegidas por un NIDS adicional para mitigar la exposición a cualquier intento de ser comprometidas desde dentro. El uso de múltiples detectores de intrusión dentro de una red sienta las bases para conseguir una arquitectura de defensa robusta.

1.1.4. Snort IDS

Snort es un IDS de red Open Source capaz de capturar y analizar tráfico en redes IP en tiempo real. Puede realizar un análisis tanto de protocolos como búsqueda y correlación de contenidos, y puede ser usado para detectar una gran variedad de ataques y código malicioso como buffer overflows, análisis sigilosos de puertos, ataques CGI, peticiones SMB, intentos de

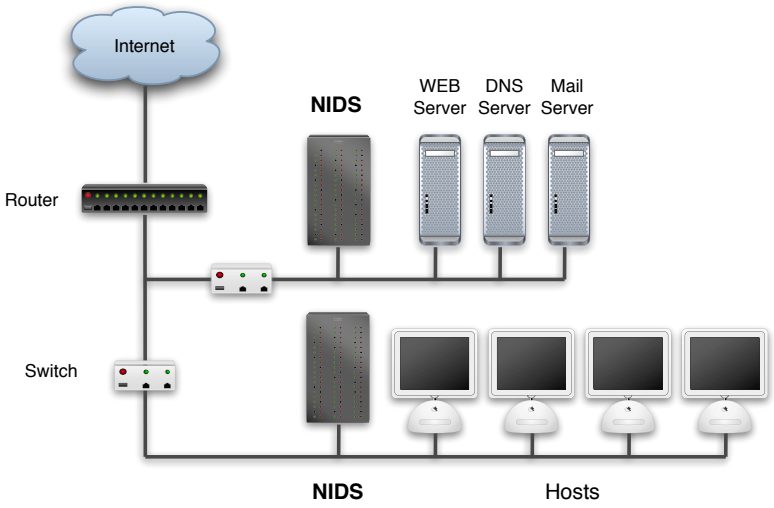


Figura 1.1: Ejemplo de arquitectura de red con detección de intrusión.

identificación del sistema operativo mediante fingerprinting, etc. Por todo ello, Snort se ha convertido muy rápidamente en la herramienta estándar para detección de intrusión.

Snort puede ser configurado en tres modos de funcionamiento: *Sniffer*, capturador de paquetes (*packet logger*) y como detector de intrusión de red (NIDS). El modo sniffer simplemente lee los paquetes de la red y los muestra de forma continuada por la consola, el modo *packet logger* permite almacenar los paquetes en el disco, y el modo detector de intrusión es el más complejo y configurable, permitiendo a Snort analizar el tráfico de red buscando correspondencias con un conjunto de reglas definidas por el usuario y realizar una entre varias acciones dependiendo de lo observado. Además de las *Community Signatures* y las firmas VDB incluidas en el paquete de Snort por el equipo de Sourcefire, es posible, para cada usuario, escribir su propio conjunto de reglas con el objetivo de adaptar el comportamiento del NIDS a las necesidades particulares de la red. Esta posibilidad proporciona una inmensa flexibilidad al motor de Snort, permitiendo adaptarlo a las necesidades de seguridad de una red específica por especiales que sean. Existen además varias comunidades online en las cuales analistas de muy alto nivel y responsables ante incidencias publican sus reglas para detectar nuevos exploits y virus recientes.

El motor de búsqueda y correlación de patrones de Snort tiene varias aplicaciones prácticas inmediatas. Por ejemplo, permite detectar hosts infectados con virus o gusanos que tienen un comportamiento característico en la red. Ya que muchos gusanos modernos se dispersan

escaneando previamente Internet y atacando hosts que consideran vulnerables, las reglas pueden ser escritas para identificar tanto este tipo de análisis o para el mismo intento de explotar la vulnerabilidad. Aunque no se considera trabajo del IDS el limpiar las máquinas infectadas, si que puede ayudar a identificar a este tipo de sistemas y puede servir, además, como confirmación de que la limpieza se ha hecho correctamente si se continúan buscando este tipo de patrones una vez que se ha realizado la limpieza y no se obtienen resultados.

Snort también posee firmas que permiten identificar el comportamiento de ataques de reconocimiento y herramientas de exploiting, aunque la mayor parte del tiempo, los desarrolladores tratan de escribir reglas ajustadas a exploits específicos y no a algún tipo de herramienta particular. No obstante, en algún caso, puede ser útil identificar cual es la herramienta que se esta usando para el ataque. Por ejemplo, existen reglas que permiten identificar la tendencia del escáner SolarWinds a incluir su nombre en la *payload* de sus paquetes de análisis ICMP. La gran mayoría de los exploits que se incluyen en herramientas muy populares, como Metasploit, tienen firmas que los identifican, permitiendo que puedan ser detectados por su comportamiento en la red [10].

1.2. Objetivos del presente proyecto

A la hora de implantar un IDS para proteger una red podremos medir la calidad de su funcionamiento, de forma general, en función de dos términos: rapidez para detectar nuevas amenazas y capacidad para detectar un mayor numero de ellas.

La rapidez de un IDS ante la publicación de nuevas vulnerabilidades dependerá, en la mayoría de los IDSs de red basados en reglas, de la pronta disponibilidad de una regla capaz de detectar el nuevo ataque correspondiente al fallo recién publicado. Esta regla podrá ser creada por un usuario de la comunidad o por el equipo de desarrollo del IDS, pero en cualquier caso, de nada serviría detectar un gran número de ataques si existe una ventana de vulnerabilidad grande. Por otra parte, parece evidente que cuanto mayor sea el número de ataques detectados por el motor de reglas, más eficaz y seguro será el IDS. Ambas condiciones representan dos buenos parámetros de medida para evaluar la calidad de un IDS.

En este proyecto se van a intentar medir dichas características de un IDS open source de amplia difusión. En un primer momento se va a realizar un estudio sobre el retardo de los procesos de actualización de las reglas de Snort en contraposición con los retardos de actuali-

zación de los diversos identificadores que suelen permitir conocer cuando se ha publicado una regla y los plugins de un sistema detector de vulnerabilidades, generalmente utilizado para detectar fallos en los sistemas y verificar el correcto funcionamiento del IDS.

De forma habitual, se utilizan sistemas de análisis de vulnerabilidades para probar la capacidad del IDS de detectar los diversos ataques o los distintos fallos, de ahí, el interés en conocer cuál es el retardo relativo existente entre la actualización de ambos sistemas. Los identificadores nos permitirán conocer el retardo de aparición de la regla respecto a la publicación de la vulnerabilidad, pero si se requiere verificar la detección del IDS, será el escáner el que nos permita comprobarlo de forma práctica y un retardo mayor en el IDS implicaría una falsa sensación de seguridad.

En una segunda parte de este proyecto y una vez analizados los retardos en los procesos de actualización, se tratará de analizar la respuesta del IDS ante ataques reales en un intento de verificar, desde otra perspectiva, la calidad de su respuesta. Para ello, se usarán herramientas ampliamente utilizadas tanto en ataques como en procesos de auditoría y se analizará la detección del IDS desde la perspectiva de un atacante al pasar por las distintas fases en el intento de vulnerar un sistema. Se probará la respuesta del IDS ante ataques de reconocimiento o escáner de puertos, primer paso en cualquier identificación de un sistema objetivo, se comprobará la capacidad de un VDS para verificar la respuesta del IDS y se analizarán los resultados obtenidos y, por último, se utilizará un framework de explotación, de uso sencillo pero muy efectivo, que permitirá conocer la respuesta de Snort ante ataques realizados con exploits reales.

A través del estudio y todas las pruebas de concepto realizadas, este proyecto tratará de profundizar en el funcionamiento de un sistema de detección de intrusión, comprender y analizar su respuesta, y verificar la calidad de su funcionamiento desde distintas perspectivas.

Capítulo 2

Gestión de Proyecto

2.1. Introducción

En este capítulo se presentan los procedimientos y mecanismos puestos en práctica para garantizar una adecuada consecución de los objetivos del proyecto. Se muestran los elementos de gestión y análisis de riesgos que han permitido cumplir con los requisitos impuestos tanto al estudio, de carácter más teórico, como a todas las pruebas de concepto realizadas.

Se indican, además, los recursos materiales necesarios para la realización del proyecto con sus principales características. Éstos se tendrán en cuenta, tras el establecimiento de la planificación, para realizar un cálculo de los costes y beneficios del presente proyecto.

2.2. Gestión de Riesgos

Este apartado pretende definir los riesgos más importantes que podrían afectar tanto a la planificación como a la consecución de los objetivos del proyecto y sus posibles mecanismos de control.

La tabla 2.1 muestra los que han sido identificados como riesgos más probables y que pueden tener un impacto mayor en el desarrollo del proyecto.

ID	Riesgo	Impacto	Mecanismo de Control
R001	Interrupción temporal debido a otras actividades de mayor prioridad (exámenes).	Elevado	M001
R002	Dificultades asociadas a la adecuación y configuración de los sistemas disponibles.	Medio	M002
R003	Pérdidas de información y/o configuraciones debido a fallos en los sistemas utilizados.	Muy Elevado	M003
R004	Inadecuaciones entre los objetivos de las pruebas realizadas y las especificaciones iniciales.	Medio	M004

Tabla 2.1: Riesgos con mayor probabilidad de aparición durante el desarrollo del proyecto.

La tabla 2.2 muestra las posibles soluciones y mecanismos a tener en cuenta para minimizar los riesgos descritos anteriormente.

ID	Mecanismo de Control	Riesgo Asociado
M001	Asignación de una franja de tiempo específico diaria para tareas relacionadas con el desarrollo del proyecto.	R001
M002	Fase previa de documentación y búsqueda de recursos tanto para configuración de sistemas y equipos como para uso de las herramientas.	R002
M003	Realización diaria de backups en diversos formatos y medios.	R003
M004	Reuniones periódicas con el tutor para redefinición de objetivos y adecuación constante de estrategias.	R004

Tabla 2.2: Mecanismos de control para los principales riesgos del proyecto.

2.3. Gestión de Recursos

En este apartado se describen los medios materiales y recursos utilizados para llevar a cabo este proyecto. Las siguientes tablas describen cada uno de los equipos, dispositivos y recursos.

ID	Concepto	Descripción
MR001	Ordenador portátil Apple Mac Book Pro Unibody	▪ Procesador Intel core 2 Duo 2.4GHz ▪ Memoria RAM 2Gb ▪ Disco duro 250Gb ▪ Sistema Operativo OSX 10.5 Leopard

Tabla 2.3: Recurso Material MR001.

ID	Concepto	Descripción
MR002	Ordenador portátil Acer Aspire 5020	▪ Procesador AMD 64 1.6Ghz ▪ Memoria RAM 1Gb ▪ Disco duro 150Gb ▪ Sistema Operativo Linux Ubuntu 9.04

Tabla 2.4: Recurso Material MR002.

ID	Concepto	Descripción
MR003	Monitor Flatron LG L206WU	▪ 21.5 Pulgadas ▪ Conexión DVI

Tabla 2.5: Recurso Material MR003.

ID	Concepto	Descripción
MR004	Wireless Router D-Link DI-524	▪ 5 Interfaces Ethernet ▪ 1 Interfaz ADSL/Cable modem

Tabla 2.6: Recurso Material MR004.

ID	Concepto	Descripción
MR005	Conexión Cable/ADSL	▪ Velocidad 2 Mbits ▪ Proveedor ONO Cable

Tabla 2.7: Recurso Material MR005.

ID	Concepto	Descripción
MR006	Software VMWare Fusion	▪ Software de virtualización ▪ Desarrollado por VMware, Inc.

Tabla 2.8: Recurso Material MR006.

2.4. Planificación

En esta sección se muestran la planificación inicial, la planificación real y se realiza una comparación y un análisis entre ambas para identificar las causas y factores que han intervenido en la temporalidad final del proyecto.

2.4.1. Planificación Inicial

El siguiente diagrama de Gantt muestra la planificación inicial del proyecto.

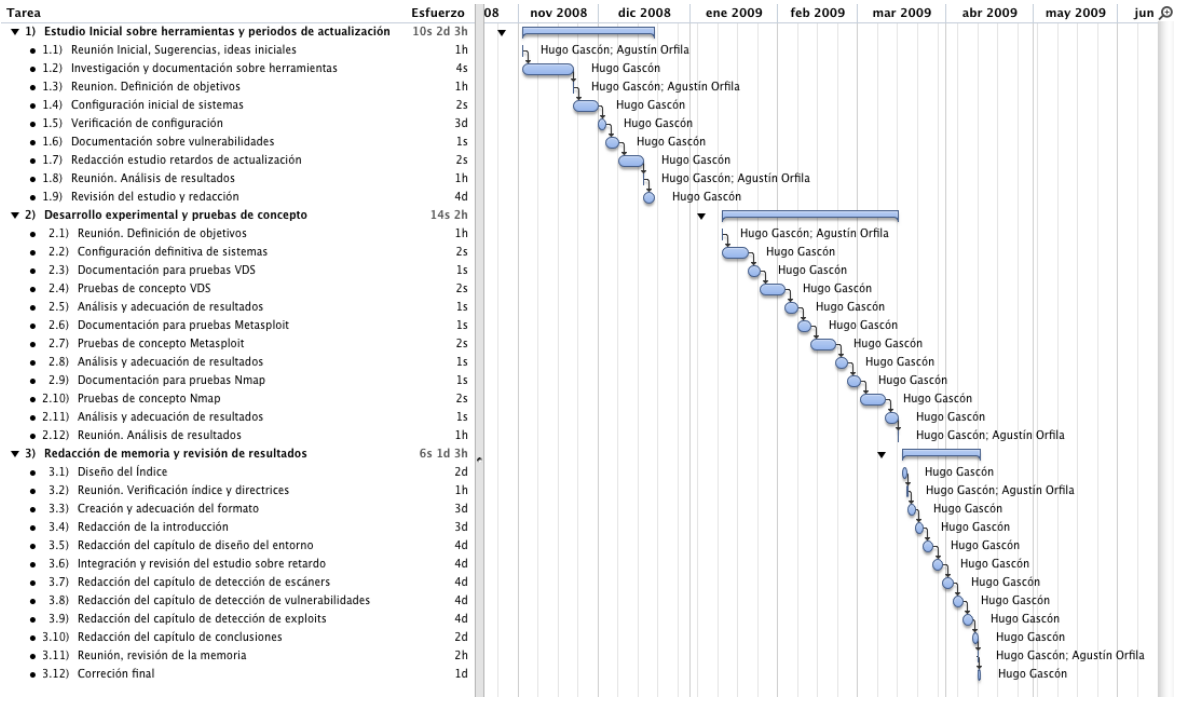


Figura 2.1: Diagrama de Gantt con planificación inicial.

El proyecto ha sido planificado con una estimación de 6 meses de duración y jornadas de trabajo de 5 horas, además, se han tenido en cuenta las vacaciones de Navidad Esto supone una planificación optimista si se tiene en cuenta que supone más de media jornada laboral de dedicación íntegra al proyecto.

2.4.2. Seguimiento real

El siguiente diagrama de Gantt muestra la planificación real del proyecto.

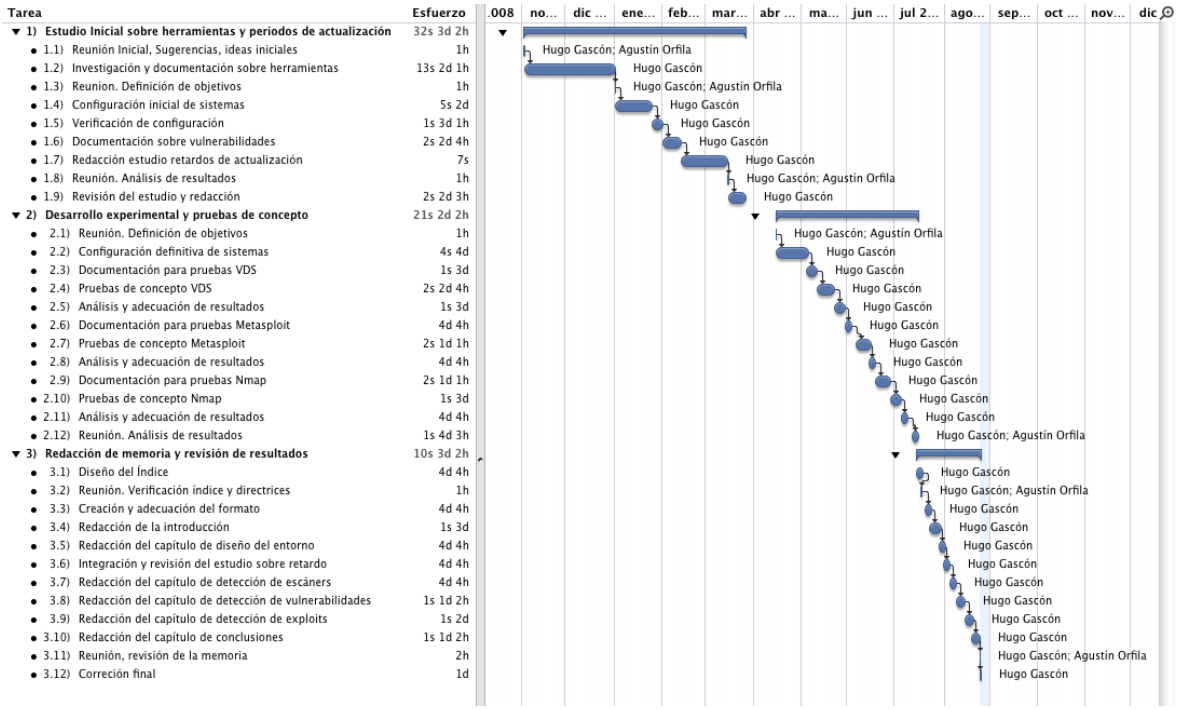


Figura 2.2: Diagrama de Gantt con planificación real.

La duración real del proyecto ha sido de 10 meses, lo cual ha supuesto un incremento del 66,6 % sobre el tiempo planificado inicialmente.

2.4.3. Conclusiones de la planificación y el seguimiento

El incremento de tiempo en la planificación inicialmente realizada lleva a la necesidad de analizar las causas del retraso en el desarrollo del proyecto, algunas de ellas recogidas en el cuadro de riesgos probables.

Podemos concluir que las siguientes son las principales causas del retraso en la planificación:

- El tiempo de estudio empleado para los exámenes ha influido considerablemente en el trabajo, reduciendo el tiempo diario dedicado al desarrollo del proyecto durante los meses de diciembre y enero, y de forma similar en mayo y junio. La finalización de las clases durante el verano y la completa dedicación al proyecto ha permitido acelerar el trabajo para su finalización antes del siguiente periodo de exámenes.

- Se han subestimado en términos de esfuerzo y tiempo muchas de las tareas planificadas, especialmente las relacionadas con la configuración y el funcionamiento de los sistemas. La ausencia de conocimientos previos acerca de las herramientas utilizadas y la necesidad de diseñar el entorno de pruebas desde cero ha requerido mucho más tiempo del planificado previamente. Durante este proceso han surgido problemas que ha sido necesario resolver de forma satisfactoria para conseguir un entorno de pruebas adecuado y tanto la documentación como el aprendizaje llevado a cabo, han supuesto un retraso considerable sobre la planificación inicial.

2.5. Costes y Presupuesto

En este apartado se va a intentar aproximar el coste del proyecto en términos económicos. Si bien tras una evaluación de los costes la evaluación de los beneficios sería una conclusión directa, en este caso resultaría muy difícil evaluar el beneficio real ya que nos encontramos ante un proyecto de investigación, distinto de un desarrollo software o algún tipo de producto. No se duda, en ningún caso, del beneficio aportado por este y otros trabajos de investigación, los cuales, como este, permiten plantear y responder preguntas cuyo impacto en la evolución de las tecnologías bajo estudio es indiscutible.

Presupuesto del Proyecto

La estimación asociada al coste de mano de obra de un Ingeniero Superior ha sido calculada en base a las horas que comprende el proyecto. 152 días y 6h de duración, con jornadas laborales de 5 horas a un coste de 14€/hora.

Tipo	**Concepto**	**Coste**
RRHH	Ingeniero Superior	10724€
HW	Portátil Apple MBP	1600€
HW	Portátil Acer Aspire 5020	1250€
HW	Monitor Flatron LG L206WU	106€
HW	Disco duro externo Backup	80€
HW	Wireless Router D-Link DI-524	49€
HW	PConexión Cable/ADSL	50€/mes
HW	3 Latiguillos UTP5	6€
SW	VVMware Fusion 2.0 (for Mac OS X)	60.62€
SW	Snort Sourcefire	0€
SW	Tenable Nessus Scanner	0€
SW	Nmap	0€
SW	Linux Ubuntu 6.06 y 9.04	0€
SW	MySQL Server	0€
SW	Apache2 Web Server	0€
SW	ACIDBASE	0€
SW	Metasploit Framework	0€
TOTAL		14375.62€

Tabla 2.9: Tabla de Costes.

Capítulo 3

Diseño del Entorno de Pruebas

En este capítulo se explicará detalladamente como se ha configurado el entorno de pruebas, tanto los diferentes equipos que han permitido simular un entorno de red adecuado para los análisis que se han llevado a cabo, como la configuración del software utilizado para los mismos.

3.1. Arquitectura de Red

El primer paso a seguir en el diseño del entorno de pruebas, es definir adecuadamente cuál será la arquitectura de red que se va a utilizar durante el proceso de análisis. Para ello, se han identificado los requisitos y se ha intentado, a partir del equipo disponible, diseñar un entorno modelo basado en una arquitectura virtual sobre la que sea posible realizar pruebas con servicios vulnerables y técnicas de ataque.

3.1.1. Red Física

Durante los últimos años y con la aparición de software que permite disponer de máquinas virtuales a un coste reducido, la mayoría de estudios relacionados con la seguridad se realizan a partir de sistemas virtuales, los cuales permiten ser configurados con servicios vulnerables en entornos controlados, que de encontrarse en un entorno de red física real, podrían conducir a resultados no deseados o incluso ser víctimas de ataques durante el proceso de estudio.

De este modo, se han considerado dos necesidades fundamentales a la hora de definir el sistema: conseguir un esquema de red vulnerable aislada del exterior y en un entorno controlado donde estén presentes los tres tipos de entidades que van a intervenir en el modelo

simplificado de detección de intrusión bajo estudio. Estos sistemas son: un atacante, una víctima y el detector de intrusión.

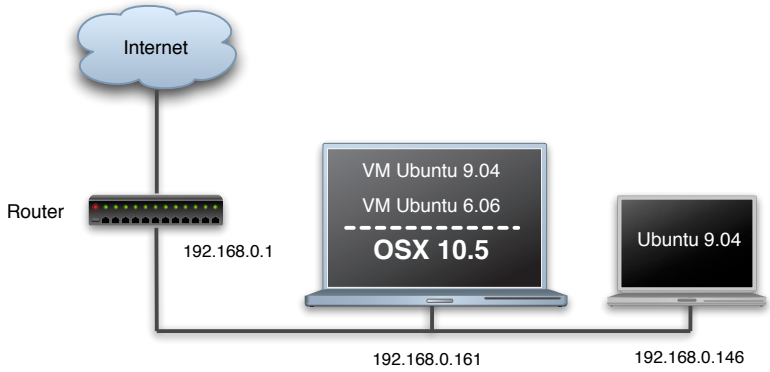


Figura 3.1: Arquitectura de red física.

La figura 3.1 representa el diseño físico real que se ha utilizado. En ella distinguimos los siguientes equipos de red:

- 192.168.0.146: El equipo portátil .146 representa al atacante. En este equipo se ha instalado un sistema operativo Linux Ubuntu Linux 9.04 y se ha actualizado completamente. Posteriormente se han instalado los programas NMap, Nessus y el framework Metasploit. Es un sistema que pretende equipararse con un equipo actual, usado en auditoria o por un atacante malicioso con la intención de vulnerar otro equipo de la red. Aunque dichas herramientas bien podrían haberse instalado en un sistema Microsoft Windows, se ha preferido el uso de Linux por estar disponible de forma abierta, y por su versatilidad y sencillez para ser configurado a bajo nivel. En cualquier caso, las herramientas de prueba están disponibles para una gran variedad de sistemas, incluidos Windows, Linux y OSX.

- 192.168.0.161: El equipo portátil .161 es el sistema más complejo de los configurados en toda la red. El hardware es un Apple Mac Book Pro sobre el que corre el sistema operativo OSX en su versión 10.5. En este sistema se ha contado con la ayuda del software de virtualización VMware Fusión, el cual ha permitido instalar sobre él dos nuevos sistemas operativos, Ubuntu Linux 9.04 y Ubuntu Linux 6.06, corriendo de forma virtual y que representan al IDS y a la víctima respectivamente. Ambas máquinas

virtuales han sido configuradas en puente a través del software de virtualización. Esto implica que el sistema *host*, es decir OSX, es totalmente transparente y ambas máquinas virtuales funcionan como equipos conectados directamente a la red como se mostrará a continuación. La elección del equipo Apple como soporte para las máquinas virtuales ha sido considerada atendiendo a su potencia y fiabilidad, ya que desde los puntos de vista tanto de computación como de memoria, soporta tres sistemas operativos de forma simultánea con sus correspondientes operaciones hardware. La figura 3.2 muestra la ventana de inicialización de las máquinas virtuales.

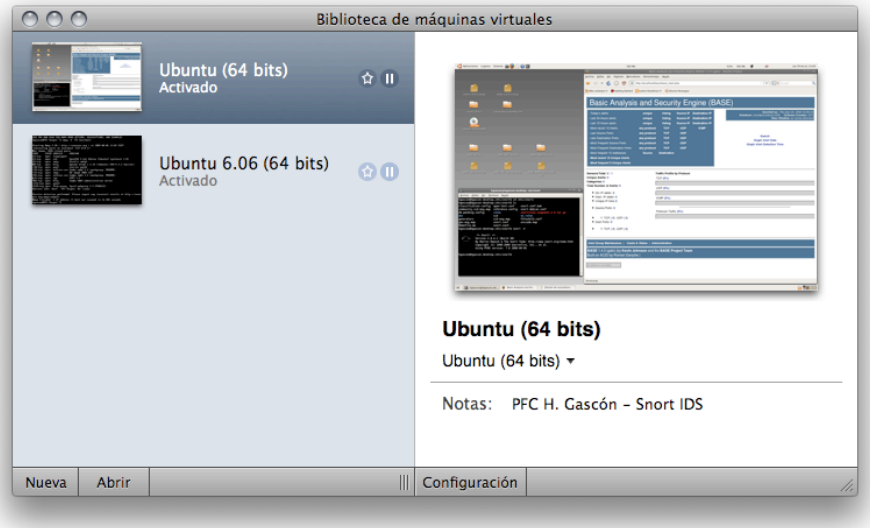


Figura 3.2: Software de virtualización VMware Fusion.

- 192.168.0.1: Por último, el sistema .1 es un router D-Link con una conexión ADSL al exterior. Si bien, este equipo permite definir la estructura de la red, su conexión con Internet resulta irrelevante para el entorno de pruebas, ya que todas las transmisiones bajo estudio que se han realizado han sido entre equipos de la red.

3.1.2. Red Virtual

Una vez configurados los equipos adecuadamente, podemos prescindir del modelo físico y centrarnos en el modelo virtual de la red, con el que podremos trabajar como si dispusiéramos realmente de tales equipos en la configuración indicada en la figura 3.3. El equipo .147 mantiene la configuración explicada previamente.

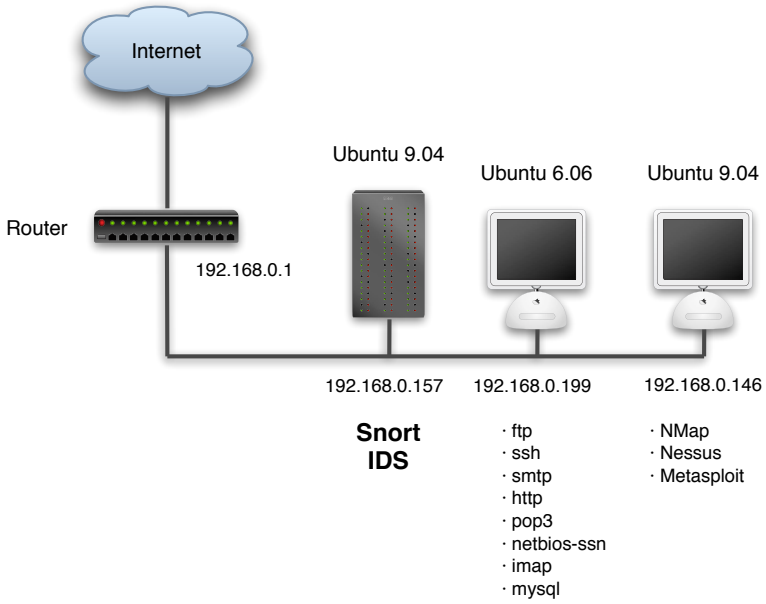


Figura 3.3: Arquitectura de red virtual utilizada como entorno de pruebas.

- 192.168.0.199: El sistema virtual .199 se corresponde con un sistema Ubuntu Linux 6.06. La elección de esta versión del sistema operativo atiende a varias razones: en primer lugar, es una versión publicada en 2006 y calificada como LTS (Long Term Support, Soporte a Largo Plazo), ya que es la primera versión de Ubuntu lo suficientemente estable como para ofrecer soporte a 3 años en la versión de escritorio y a 5 en los servidores, es decir, una razón importante para considerar que ha sido instalada en un numero considerable de empresas durante los últimos años. En segundo lugar, posee una antigüedad suficiente como para poder encontrar servicios con vulnerabilidades de forma sencilla y desde los puntos de vista académico y empresarial, nos interesa conocer la respuesta del detector de intrusión respecto a sistemas Linux. El repositorio situado en:

  http://security.ubuntu.com/ubuntu/

  ha permitido instalar una serie de servicios vulnerables y suficientemente comunes para ser utilizados como objeto de prueba. La figura 3.4 muestra la salida por consola de un listado de los servicios vulnerables y disponibles en la máquina objetivo.

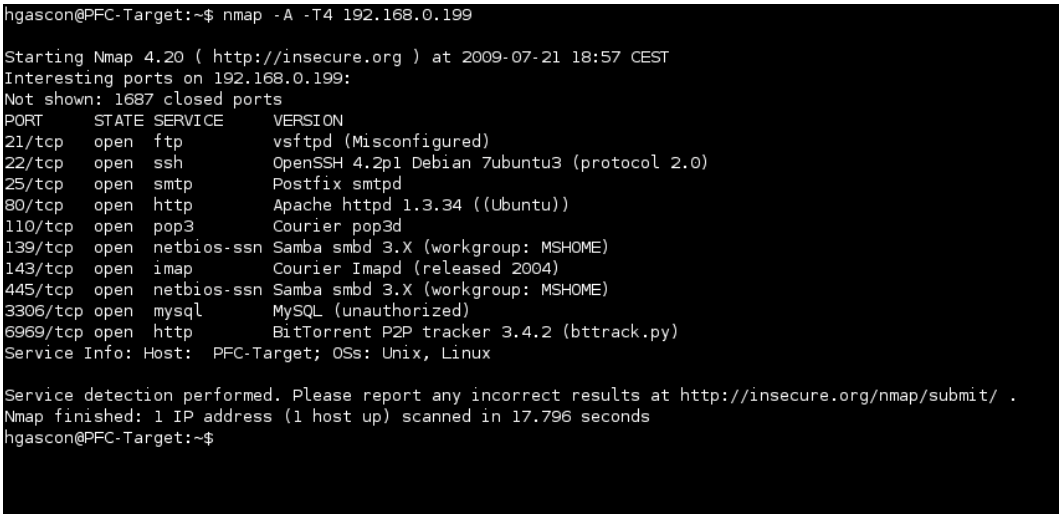


Figura 3.4: Servicios vulnerables instalados en el equipo 192.168.0.199.

- 192.168.0.157: Es el equipo virtual más importante de todos ya que es donde se encuentra configurado el sistema de detección de intrusión. Para servir de soporte al IDS se ha seleccionado la última versión del sistema Ubuntu Linux, 9.04, cuya configuración puede asemejarse a la de un equipo actual utilizado con este mismo fin. Es un sistema completamente actualizado en el que se han instalado el IDS Snort y el sistema de visualización y gestión de alertas BASE.

3.2. Configuración

3.2.1. Snort

A pesar de que Snort puede ser configurado en diferentes modos, en este proyecto se ha estudiado su comportamiento como NIDS. Para conseguir que Snort se encuentre completamente instalado y configurado el primer paso ha sido descargar el paquete de la dirección:

http://dl.snort.org/snort-current/snort-2.8.4.1.tar.gz

Una vez descargado, se ha descomprimido y compilado con soporte para la base de datos *mysql* mediante los comandos:

```
$> tar -zxvf snort-2.8.4.1.tar.gz
$> sudo ./configure --with-mysql
```

```
$> sudo make
$> sudo make install
```

El siguiente paso y uno de los mas complejos si no se conoce adecuadamente Snort, es la edición de su fichero de configuración *snort.conf*, el cual se ha modificado de la siguiente manera:

```
$> sudo vi /etc/snort/snort.conf
```

Una vez abierto con el editor vi, se han configurado las siguientes variables y se han activado todos los ficheros de reglas disponibles:

```
var HOME_NET [192.168.0.199/32]
var EXTERNAL_NET any
output database: log, mysql, user=snort password=snort dbname=snort host=localhost
```

Respecto a las reglas de detección utilizadas por el motor de Snort, cabe destacar que la empresa Sourcefire y su equipo de desarrollo *Sourcefire Vulnerability Research Team (VRT)* distribuyen las nuevas reglas oficiales bajo un acuerdo de licencia que permite estudiar y modificar dichas reglas pero restringe la redistribución comercial. La publicación está basada en dos modalidades de suscripción:

- Suscriptores: Acceso en tiempo real a actualizaciones certificadas del equipo de desarrollo VRT. Se requiere una suscripción pagada.
- Usuarios registrados: Usuarios registrados de Snort.org pueden descargar y usar las reglas del equipo VRT sin cargo 30 días después de su publicación inicial.

Las reglas utilizadas para este proyecto son las publicadas para usuarios con suscripción el 16 de Mayo de 2009 y para usuarios registrados un mes después, el 16 de Junio de 2009. Se entiende, por tanto, que la respuesta del IDS es la que tendría un sistema actualizado por usuarios de pago a 16 de Mayo de 2009 y por usuarios registrados, a 16 de Junio de 2009.

3.2.2. BASE

Acidbase es un interfaz web por el cual podemos visualizar fácilmente los registros almacenados en la base de datos y nos da la posibilidad de crear o modificar reglas nuevas para su análisis posterior o realizar consultas a la base de datos. Es necesaria la instalación de un servidor web con soporte PHP que permita acceder a la interfaz por medio del navegador. En este caso y partir de los repositorios, se han instalado los siguientes paquetes:

- acidbase 1.3.9-2
- php5 5.2.6.dfsg.1-3ubuntu4.1
- apache2 2.2.11-2ubuntu2.1

Es necesaria una base de datos en la que Snort pueda escribir los eventos y alertas que va generando. En este caso se ha escogido *MySQL*. Tanto Acidbase como snort soportan diferentes bases de datos como MySQL, postgresql, odbc, mssql y oracle. El paquete instalado es:

- mysql 5.1.30really5.0.75-0ubuntu10.2

El último paso antes de poder iniciar Snort es la creación de una base de datos mediante la serie de comandos:

```
$> mysql -u root -p
mysql> CREATE DATABASE snort;
mysql> grant CREATE, INSERT, SELECT, UPDATE on
snort.* to snort@localhost;
mysql> grant CREATE, INSERT, SELECT, UPDATE on
snort.* to snort;
mysql> SET PASSWORD FOR
snort@localhost=PASSWORD('snort');
mysql> flush privileges;
mysql> quit
```

Es necesario crear las tablas que usará Snort en la base de datos mediante una plantilla incluida en el paquete de instalación de Snort:

```
$> cd /usr/share/doc/snort-mysql/
$> zcat create_mysql.gz | mysql -u snort -h localhost -p snort
```

Por último es necesario configurar BASE con los parámetros de la base de datos:

```
$> sudo vi /etc/acidbase/database.php
$alert_user='snort';
$alert_password='UN_PASSWORD';
$basepath='';
$alert_dbname='snort';
$alert_host='localhost';
$alert_port='3306';
$DBtype='mysql';
```

Mediante los siguientes comandos es posible indicar a BASE la versión de apache a utilizar y arrancar el servidor web:

```
$> sudo dpkg-reconfigure acidbase
$> sudo /etc/init.d/apache2 restart
```

El último paso es iniciar Snort, lo cual puede hacerse en modo demonio o desde la consola. Para el análisis llevado a cabo se ha preferido el arranque manual, que permite mayor control sobre las opciones y una parada sencilla en caso necesario:

```
$> sudo snort -i eth0 -c /etc/snort/snort.conf
```

La figura 3.5 muestra la interfaz BASE tal y como se presenta en el navegador web una vez que Snort ha generada algún tipo de alerta.

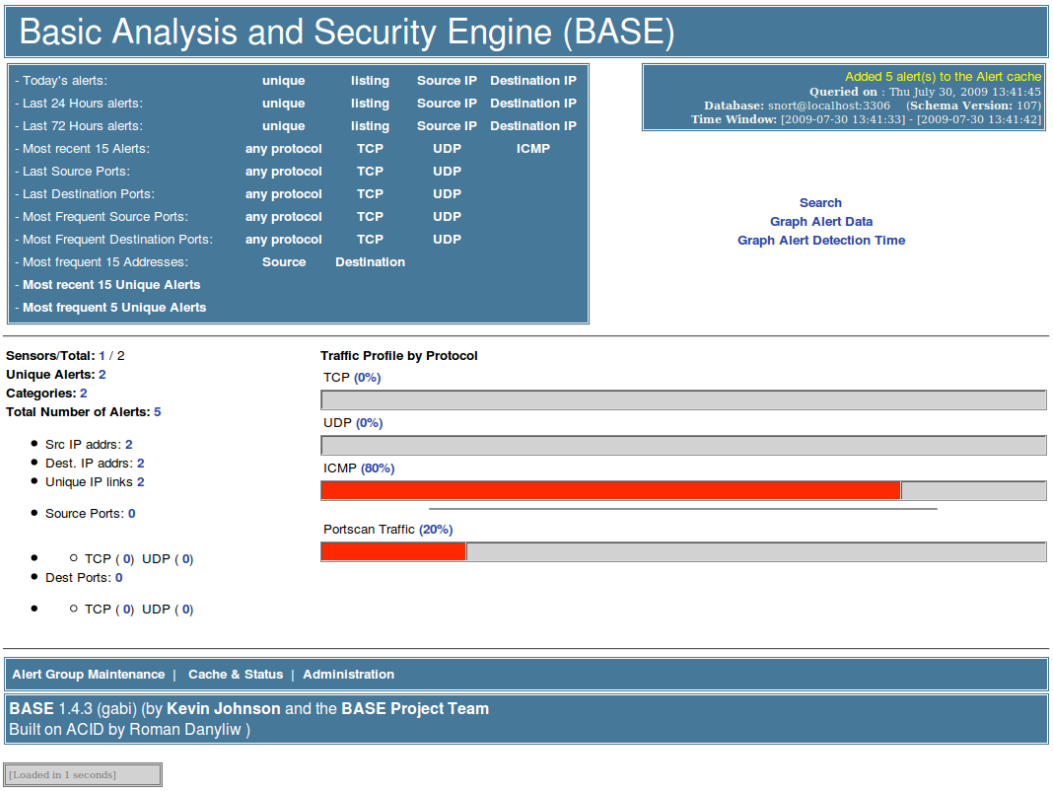


Figura 3.5: Interfaz BASE (Basic Analysis and Security Engine).

Capítulo 4

Retardos de Actualización

4.1. Introducción

El objetivo de este capítulo es realizar un análisis teórico del retardo del proceso de actualización del IDS Snort respecto de un software de análisis de vulnerabilidades como Nessus. Para ello se han utilizado como referencia los identificadores CVE y Bugtraq y sus fechas de actualización. Ambos identificadores, aunque asociados a organizaciones diferentes, permiten identificar de forma unívoca las vulnerabilidades a las que hacen referencia, así como los detalles asociados a las mismas.

Como resultado de este análisis se muestran tres gráficas de la distribución del retardo de actualización respecto de los indicadores CVE, Bugtraq y la fecha de publicación de los plugins de Nessus para cada vulnerabilidad analizada. Por último, se presenta una tabla del retardo de la actualización de la vulnerabilidad en la aplicación en comparación con la actualización de reglas de Snort y los plugins de Nessus, es decir, se calcula la ventana de vulnerabilidad de la aplicación y se compara con la ventanas vulnerables de Snort y Nessus[5] [15][6][8][2].

Como pasos de la metodología seguida y para la compresión del trabajo realizado, es necesario tener en cuenta los siguientes puntos:

- Para la realización del estudio del retardo y a diferencia de las pruebas de concepto realizadas en los siguientes capítulos, se han tenido en cuenta las reglas de nueva creación introducidas en Snort desde principios del año 2009 y publicadas hasta la fecha 30/03/2009.

- Se han analizado las reglas de nueva creación para cada actualización del paquete de reglas y para un nivel de riesgo establecido por el equipo de Snort como "high". Podría hacerse también una comparativa entre reglas ya existentes y que son actualizadas para detectar una actualización de la vulnerabilidad, sin embargo, analizando reglas de nueva creación, nos aseguramos de que dicha regla pretende ser una respuesta a la última actualización de una vulnerabilidad antigua o de nueva aparición.

- Para poder establecer el retardo en la aparición de la regla se indica la última fecha de actualización de la vulnerabilidad anterior a la aparición de la regla de Snort. Si existe una actualización posterior a la fecha de publicación de la regla de Snort, ésta no se considera relevante para el estudio y se indica con un asterisco (*) en el nombre de la misma. Es posible que en actualizaciones posteriores de las reglas de Snort, hayan aparecido reglas para detectar dicha actualización de la vulnerabilidad, pero si son posteriores a la fecha de aparición del paquete de reglas utilizado en el estudio, no son, por tanto, útiles para éste.

- Si la fecha de creación del identificador o del plugin Nessus es posterior a la aparición de la correspondiente regla de Snort, el número de días de desfase aparece en negativo. Esto puede ser interpretado como una ventaja desde el punto de vista de las actualizaciones de Snort, y es así como debe ser entendido en las gráficas de distribución del retardo.

- El objeto de utilizar como referencia los identificadores CVE y Bugtraq es identificar cuál es el retardo real que existe entre la aparición de la regla y el plugin Nessus. Es posible que parezca existir un retardo muy elevado entre el plugin y la regla, cuando es en realidad la regla la que se ha actualizado recientemente (como muestran los identificadores) y ser sin embargo el plugin, el que permanece desfasado.

- Los identificadores marcados con **, son aquellos que se encuentran desfasados. En este caso, la actualización de la regla Snort puede ser comparada con la actualización de algún otro de los identificadores para una vulnerabilidad concreta.

- En algún caso, el indicador aparece marcado como ***. Se ha tenido en cuenta que la regla de Snort tan solo hace referencia a un identificador, ya sea CVE o bugtraq, y que de éste tan solo se dispone de la fecha de creación y la última fecha de actualización, siendo la primera muy temprana y la segunda posterior a la fecha de publicación de la

regla de Snort, con lo que no se puede definir con certeza el retardo al que responde la publicación de la regla de Snort.

- En el caso de los plugin de Nessus que están marcados con **, se entiende que la publicación de dicho plugin tiene como origen la fecha de publicación de la vulnerabilidad, por lo que no se puede garantizar que el plugin sea capaz de testar la última actualización de la vulnerabilidad a la que Snort si es capaz de responder con dicha regla.

- Los retardos correspondientes a identificadores o plugins marcados como ** o *** dan lugar a retardos irreales, de modo que se ha asignado un valor de retardo nulo para la publicación de la regla. Esto concuerda con el hecho de que la regla se encuentra, en efecto, al día, respecto de los indicadores o el plugin correspondiente.

- El listado de vulnerabilidades utilizadas como datos, junto con sus fechas de publicación para los diferentes identificadores, Snort y Nessus, se muestran como un capítulo en el Anexo A, al final de la memoria del proyecto.

4.2. Dispersión del retardo de actualización respecto de los identificadores CVE

La base de datos CVE puede considerarse un diccionario público de vulnerabilidades y fallos conocidos de los sistemas de información. Los identificadores CVE (*Common Vulnerabilities and Exposures*) permiten el intercambio de información entre fabricantes de productos y personal técnico especializado, siendo un indicador de la cobertura que un determinado producto presenta desde el punto de vista de la seguridad. Los identificadores CVE son códigos únicos que facilitan la identificación de cada vulnerabilidad. Cada CVE incluye un numero, por ejemplo CVE-1999-0067, indicando la entrada o el estado como candidato, una descripción breve y las referencias pertinentes de fabricantes de producto o documentación técnica sobre el fallo. Son habitualmente utilizados por fabricantes e investigadores como un método para enlazar las vulnerabilidades con otros repositorios que también utilizan identificadores CVE.

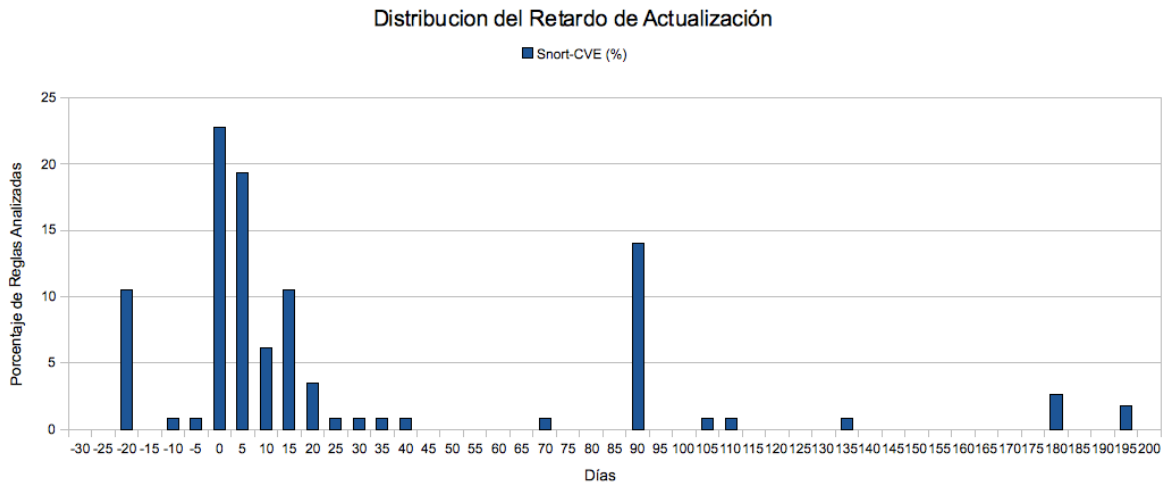


Figura 4.1: Dispersión del retardo de actualización entre identificadores CVE y reglas de Snort.

La gráfica 4.1 muestra el perfil de la dispersión del retardo de actualización de los indicadores CVE para las vulnerabilidades analizadas y la aparición de las reglas Snort que las identifican.

Estadístico	**Días**
Media	27,24
Desviación Típica	49,5
Máximo	192
Mínimo	-20

Tabla 4.1: Médidas estadísticas del retardo Snort-CVE.

La tabla 4.1 muestra algunos parámetros estadísticos de la dispersión del retardo calculada.

4.3. Dispersión del retardo de actualización respecto de los identificadores BugTraq

BugTraq es una lista de correo moderada basada en la filosofía del *full disclosure*, cuyo objetivo es la discusión detallada y la publicación de nuevas vulnerabilidades con información detallada sobre las mismas y como explotarlas y solucionarlas. La lista de correo se encuentra

alojada en la dirección:

http://www.securityfocus.com/archive/

donde también está disponible la *SecurityFocus Vulnerability Database*:

http://www.securityfocus.com/bid

Esta base de datos proporciona a los diferentes profesionales e investigadores un listado actualizado de vulnerabilidades existentes para diferentes plataformas y servicios. Cada una de ellas se encuentra asociada a un identificador BugTraq. Muchos de las descripciones incluyen el indicador CVE que hace referencia a la misma vulnerabilidad, de forma que es posible establecer una correlación entre las distintas bases de datos.

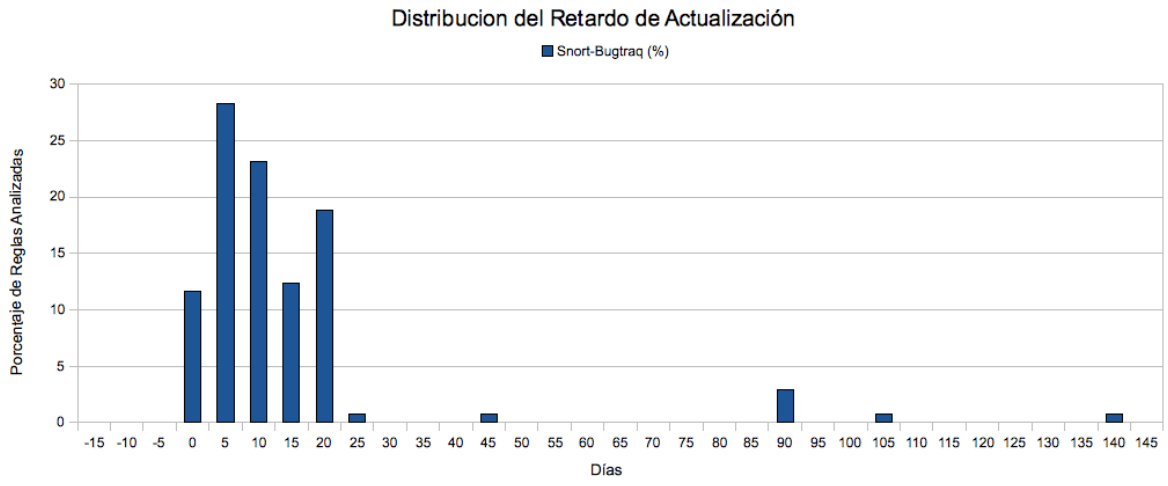


Figura 4.2: Dispersión del retardo de actualización entre identificadores BugTraq y reglas de Snort.

La gráfica 4.2 muestra el perfil de la dispersión del retardo de actualización de los indicadores BugTraq para las vulnerabilidades analizadas y la aparición de las reglas Snort que las identifican.

La tabla 4.2 muestra algunos parámetros estadísticos de la dispersión del retardo calculada.

Estadístico	Días
Media	14,24
Desviación Típica	20,16
Máximo	140
Mínimo	-2

Tabla 4.2: Médidas estadísticas del retardo Snort-Bugtraq.

4.4. Dispersión del retardo de actualización respecto de los plugins de Nessus

Nessus es un escáner de vulnerabilidades que permite auditar sistemas y verificar la existencia de servicios vulnerables en los mismos. Cada vulnerabilidad que Nessus es capaz de identificar es detectada por medio de un script o plugin. En el capítulo 6 se presenta una revisión más detallada de su funcionamiento y configuración. En este apartado se ha intentado comparar el retardo de actualización de los plugins de Nessus respecto de las reglas de Snort, ya que en muchos casos, y como se mostrará mas adelante, Nessus es utilizado para verificar un correcto funcionamiento del IDS.

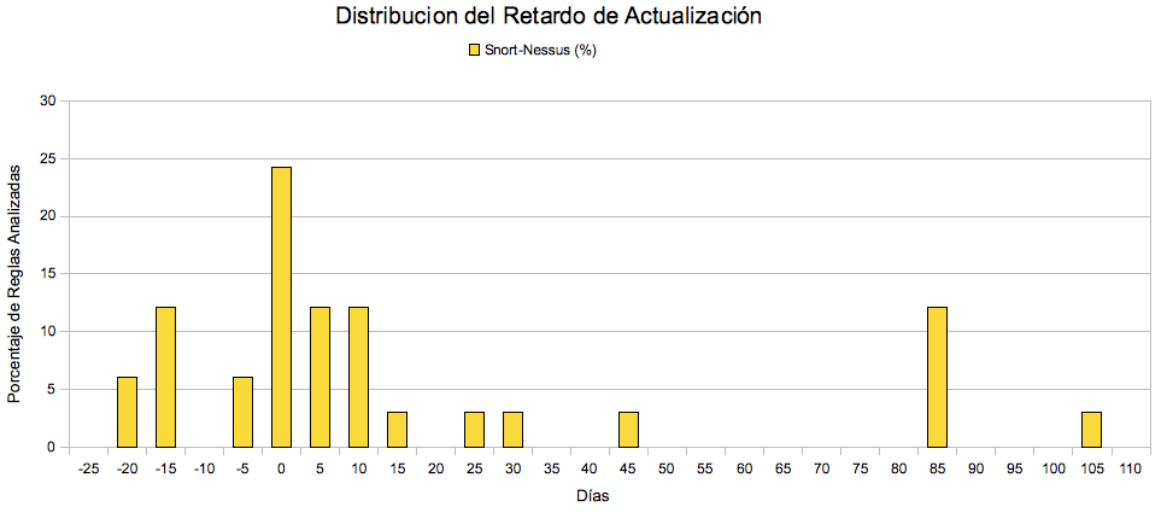


Figura 4.3: Dispersión del retardo de actualización entre plugins Nessus y reglas de Snort.

La tabla 4.3 muestra algunos parámetros estadísticos de la dispersión del retardo calculada.

Estadístico	Días
Media	15,45
Desviación Típica	33,96
Máximo	102
Mínimo	-21

Tabla 4.3: Médidas estadísticas del retardo Snort-Nessus.

4.5. Ventanas de vulnerabilidad

En este apartado, se ha analizado la ventana de vulnerabilidad existente desde la publicación de una vulnerabilidad en un software hasta la aparición de la actualización que solventa dicho fallo. Además se ha comparado la ventana vulnerable de Snort y Nessus con la ventana vulnerable del software para comprobar cuál de los tres métodos, la actualización de software, la detección por Nessus o la detección mediante el IDS, proporciona una respuesta más rápida.

En la tabla 4.4 se muestra la información relativa a una serie de vulnerabilidades seleccionadas a partir de la lista detallada en el Anexo A. Se han escogido aquellas vulnerabilidades de las que se disponía de un mayor número de datos y que permiten una comparativa realista entre Snort y Nessus.

Algunos datos a tener en cuenta sobre la información recogida en la tabla:

- Los días que se indican en la columna "Ventana Software", representan el tiempo que transcurre desde la publicación de la vulnerabilidad hasta la aparición del parche o de la nueva versión de dicho software. Un número de días negativo indica que la vulnerabilidad se difundió una vez publicada la actualización de software. El número indica la diferencia entre ambos momentos.

- Los días que se indican en la columna "Ventana Snort", representan el tiempo que transcurre desde la publicación de la vulnerabilidad hasta la aparición de una regla que sea capaz de detectar un ataque a la misma.

- Los días que se indican en la columna "Ventana Nessus", representan el tiempo que transcurre desde la publicación de la vulnerabilidad hasta la aparición de un plugin que es capaz de comprobar la existencia de ese fallo en un sistema.

- Las ventanas de Snort marcadas como '—' se corresponden con plugins de Nessus marcados como **. En estos casos, sólo se ha podido verificar la ventana vulnerable para los plugins Nessus, ya que éstos respondían a la publicación inicial de la vulnerabilidad mientras que las reglas de Snort se publicaban tras una modificación de la misma tiempo después.

- Los ventanas de Nessus marcadas como '—', corresponden a plugins de los que no se ha podido identificar la fecha de publicación.

- Para cada vulnerabilidad se ha marcado en negrita cual de los tres métodos posee la menor ventana de tiempo vulnerable.

CVE	V. Software	Snort Rule	V. Snort	Nessus Plugin	V. Nessus
2009-0771*	2	15428	23	35778	**1**
2009-1169*	**0**	15431	**0**	36045	3
2008-4769	**23**	15432	—	32080**	27
2009-0208	**2**	15380	9	35804	15
2007-4568	222	15382	—	30254**	**130**
2009-0241*	**3**	15364	44	35564	18
2009-0658*	**0**	15358	5	35821	22
2009-0658*	**0**	15356	1	35821	22
2008-5260	**-4**	15243	10	35454	1
2008-5444*	**-1**	15255	13	35374	1
2008-4006*	**-1**	15257	13	35363	—
2008-5448*	**-1**	15261	13	35363	—
2008-5276	**3**	15241	50	35068	9
2008-3641*	**0**	15186	87	34385	2
2008-4064*	**0**	15191	103	34267	1

Tabla 4.4: Ventanas de vulnerabilidad en días para actualizaciones de software, nuevas reglas para Snort IDS y nuevos scripts para Nessus VDS.

4.6. Conclusiones

El análisis llevado a cabo ha permitido poner de manifiesto, no sólo la existencia de retardos en los procesos de actualización de los diferentes identificadores y herramientas, sino también la gran variabilidad de los mismos. La consecuencia directa de este hecho es la dificultad para los distintos administradores de red y responsables de seguridad a la hora de fijar una política de seguridad adecuada, basada en actualizaciones y *parcheado* de las diferentes aplicaciones vulnerables.

Los resultados permiten aventurar la aparición de situaciones en las que Nessus es capaz de detectar vulnerabilidades mediante ataques ante los cuales el IDS Snort no genera ninguna alerta. O simplemente, la constatación de que vulnerabilidades que han sido publicadas y que son susceptibles de ser explotadas pueden permanecer indetectables para Snort durante un periodo de tiempo variable.

Esta diferencia en los procesos de actualización de las distintas herramientas, las decisiones de cada fabricante respecto a qué actualizar y con qué rapidez, da lugar a una dificultad considerable desde un punto de vista de coordinación y sugiere la necesidad de creación de un sistema centralizado que favorezca la identificación y actualización coordinada de los sistemas de seguridad involucrados.

Por otra parte, se ha comprobado como la respuesta de los desarrolladores es, en la mayoría de los casos, la más rápida en comparación con la de los responsables de las herramientas de detección de vulnerabilidades e intrusiones. Esto se debe, en parte, al acierto de muchos equipos de desarrollo a la hora de publicar las vulnerabilidades previa distribución de las versiones actualizadas del software. La ventana vulnerable de las aplicaciones es habitualmente menor, lo que permite asegurar que una política adecuada de actualizaciones en las aplicaciones debe ser el primer paso a la hora de garantizar la seguridad de los sistemas vulnerables.

Capítulo 5

Detección de Ataques de Reconocimiento

5.1. Introducción

La primera fase de prácticamente cualquier ataque a través de la red es la etapa de reconocimiento. En esta fase, un atacante intenta determinar cuáles son los protocolos y servicios soportados por el *host*. Es la etapa en la que posiblemente se llevará a cabo un escáner de puertos. Desde el punto de vista defensivo, en esta fase se asume que el atacante no tiene conocimiento a priori de los servicios disponibles o los protocolos soportados por el objetivo, en caso contrario, esta fase no sería necesaria.

Como el atacante no dispone de información previa sobre el objetivo la mayoría de las peticiones enviadas obtendrán una respuesta negativa, lo que será interpretado como la indisponibilidad, en el sistema objetivo, de dichos servicios, es decir, los puertos correspondientes se encontrarán cerrados. En la mayoría de comunicaciones legítimas de una red, las respuestas negativas desde un *host* son poco comunes, y nada comunes si se producen un número considerable de respuestas negativas consecutivas en un periodo corto de tiempo. El primer objetivo para detectar un escáner de puertos, desde el punto de vista del IDS, es monitorizar y detectar estas respuestas negativas.

Para realizar este tipo de detección compleja, donde es necesario monitorizar cada una de las conexiones que cada protocolo genera y el comportamiento de los diferentes *hosts* que intervienen en ellas, el IDS Snort cuenta con una serie de módulos denominados *preprocesadores*. Éstos son complejos complementos de código que pueden ser cargados por Snort para realizar tareas no sólo de detección de anomalías y normalización de protocolos sino también

para generar alertas independientes a partir de diversos eventos. Es importante comprender, que los preprocesadores no forman parte del motor de detección de Snort, entendido éste como el conjunto de código que implementa las reglas que se activan con Snort y realiza las comprobaciones para cada una de ellas. Es decir, los preprocesadores no están basados en reglas, son programas con código independiente, cada uno con su propia configuración, realizando cada uno una tarea independiente, pero funcionando en conjunto para ofrecer al IDS una visión lo más simplificada posible del tráfico supervisado [9].

Aunque Snort dispone de varios preprocesadores que intervienen en la detección de los ataques de reconocimiento, es uno especialmente, *sfPortscan*, el que está diseñado de forma específica para esa tarea, siendo capaz de identificar diferentes tipos de escáners y generar un número considerable de alertas según la actividad de los sistemas implicados.

El objetivo de este capítulo es analizar la respuesta del IDS ante este tipo de ataques y comprobar, en la medida de lo posible, las limitaciones o no de sus preprocesadores y en especial de sfPortscan, para generar alertas ante intentos de evitar la detección del ataque de reconocimiento. Para ello, se ha contado con Nmap, una de las herramientas mas utilizadas actualmente, convertida prácticamente en un estándar para la realización de escáners de puertos y que permite llevar a cabo de forma relativamente sencilla muchos, si no todos los tipos actuales de análisis. Además, Nmap implementa una serie de opciones cuyo objetivo es evitar la detección del escáner y es este tipo de configuración el que se ha utilizado para comprobar la respuesta de Snort en varias pruebas de concepto.

5.2. Preprocesadores en Snort

Los preprocesadores son programas independientes compilados en Snort y que realizan funciones de normalizado y supervisión del tráfico para detectar ataques de una forma más compleja que la detección por firmas, utilizada por el motor de reglas. La figura 5.1 representa el lugar que ocupan los preprocesadores en el proceso de detección llevado a cabo por Snort.

Snort posee tres preprocesadores que permiten reensamblar paquetes con información dividida en múltiples tramas. En el caso de comunicaciones sobre protocolo TCP/IP, la propia red puede fragmentar los paquetes con el objetivo de optimizar el funcionamiento del flujo. Los paquetes pueden llegar desordenados o fragmentados en tramas más pequeñas, sin embargo, esto también puede ser utilizado por un atacante para conseguir evadir la detección del IDS,

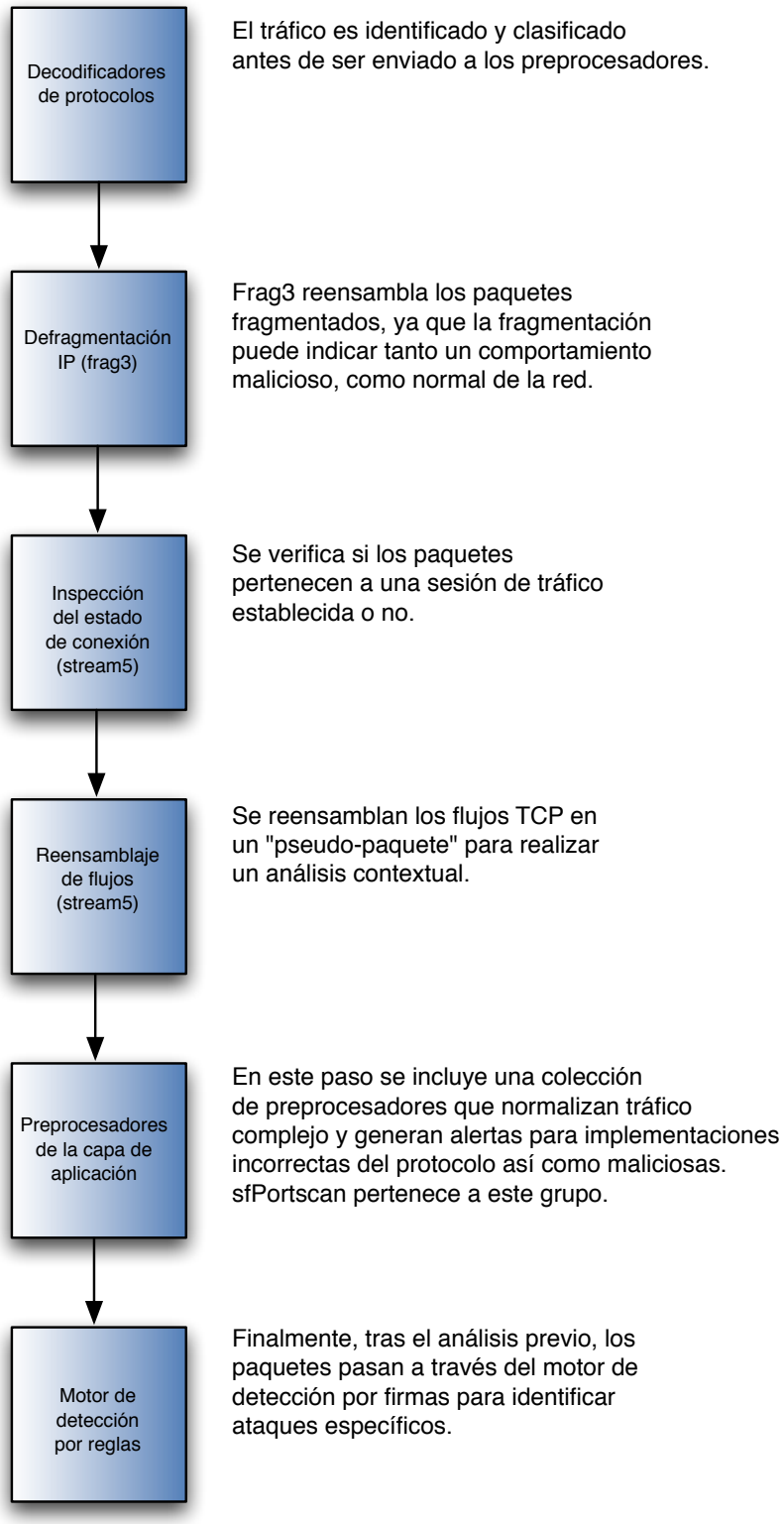


Figura 5.1: Esquema de procesamiento de tráfico por preprocesadores en Snort.

intentando ocultar el ataque mediante paquetes fragmentados que serán reensamblados en el *host* objetivo y no en un punto intermedio de la red.

Para poder realizar este tipo de operaciones que servirán de base al preprocesador sf-Portscan a la hora de detectar ataques de reconocimiento, Snort cuenta entre otros con los siguientes preprocesadores:

frag3

El preprocesador *frag3* permite implementar el concepto de IDS *"target-based"*, esto es, analizar el tráfico como el objetivo o el sistema operativo del *host* final lo haría. Diferentes sistemas operativos reensamblan los paquetes en recepción de forma diferente y se conocen métodos mediante los cuales se puede evadir la detección del IDS utilizando este hecho [16]. La función de Frag3 es, de forma general, reensamblar los paquetes de forma diferente según los sistemas que se estén defendiendo.

flow

El preprocesador *flow* es un módulo pequeño en cuanto a líneas de código pero muy importante en cuanto a funcionalidad. Permite definir y monitorizar qué sistemas intervienen en un flujo de datos, cuál es el cliente y cuál el servidor y desde qué puertos están transmitiendo y recibiendo.

stream5

El preprocesador *stream5* permite identificar errores en las sesiones TCP ya sean como consecuencia de un funcionamiento defectuoso de la red o con la intención de obtener un efecto malicioso (Resets o FINs inadecuados, etc.). Además puede guardar en memoria paquetes anteriores de un flujo para identificar, mediante reensamblaje de la sesión completa, ataques cuyo código se encuentra repartido en varias tramas. Stream5 es un preprocesador complejo con un gran número de opciones de configuración y que requiere un ajuste adecuado para optimizar su capacidad de detección.

5.3. El Preprocesador sfPortscan

El preprocesador sfPortscan es un módulo diseñado específicamente para la detección de la primera fase de un ataque. A partir de la idea de que Nmap es la herramienta por excelencia para realizar escáners de puertos, sfPortscan ha sido diseñado para identificar los diferentes tipos de análisis posibles utilizando Nmap.

La siguiente es una lista de los tipos de análisis de Nmap que sfPortscan es capaz de identificar:

- TCP
- UDP
- IP

Este tipo de alertas se generan para análisis uno a uno, es decir, realizados desde un *host* sobre múltiples puertos en otro *host*. Este es el tipo de ataque tradicional. La mayoría de las peticiones a los puertos tendrán una respuesta negativa ya que la mayoría de los *hosts* disponen de relativamente pocos servicios disponibles.

Los análisis de tipo *decoy*, como ya se ha explicado, son en esencia igual a los anteriores y también pueden ser detectados por sfPortscan:

- TCP decoy
- UDP decoy
- IP decoy

Cuando se produce un ataque de reconocimiento desde muchos sistemas hacia un objetivo en particular, esté se denomina escáner distribuido, es decir, muchos *hosts* envían peticiones a un único *host* para detectar servicios disponibles. Este tipo de escáner puede ser producido por una Botnet con el objetivo de evadir la detección de un IDS. En este tipo de escáner, las respuestas negativas serán distribuidas entre los *hosts* que realizan el ataque, de modo que para poder detectarlo sfPortscan intenta monitorizar el tráfico desde el punto de vista del *host* objetivo.

- TCP Distribuido

- UDP Distribuido
- IP Distribuido

En el caso de producirse un escáner desde un solo *host* a un elevado número de *hosts*, el tipo de ataque se denomina *portsweep*. En este caso, un solo *host* realiza peticiones al mismo puerto de diferentes máqinas. Este tipo de ataque es común cuando un nuevo exploit es publicado y el atacante está buscando un servidor específico.

- TCP Portsweep
- UDP Portsweep
- IP Portsweep
- ICMP Portsweep

En el caso de un escáner de tipo portsweep, es posible no recibir numerosas respuestas negativas. Por ejemplo, si un atacante realiza un portsweep sobre una granja de servidores web en busca del puerto 80, es posible que al estar los servidores activos, no se reciba un gran número de respuestas negativas, por lo que no se podrá identificar el ataque de este modo.

En algunos casos, pueden generarse alertas de tipo *filtered*, que indican que no se han producido errores como *ICMP unreacheables* o paquetes TCP con el bit RESET activo. También es posible que se haya configurado el sistema para no responder en caso de que se realicen peticiones a servicios inactivos. Algunos *hosts* como los NATs pueden dar lugar a alertas de este tipo al enviar un gran número de solicitudes de conexión en un periodo corto de tiempo en el caso de que la los *hosts* remotos no respondan antes de que se genere la alerta.

sfPortscan sólo genera una alerta para cada pareja de *hosts* durante el tiempo de ventana y si se trata de un alerta TCP, mostrará los puertos abiertos que han sido escaneados. En el caso de un escáner de tipo portsweep, solamente se mostrarán los puertos abiertos una vez que la alerta haya sido generada. Los eventos que se generan cuando se detecta un puerto abierto no son considerados como alertas pero sí como etiquetas dependientes de la alerta original del escáner.

Las siguientes son las posibles combinaciones de análisis que pueden ser identificados por sfPortscan y generar alertas de tipo filtrado:

- TCP Filtrado
- UDP Filtrado
- IP Filtrado
- TCP Filtrado Decoy
- UDP Filtrado Decoy
- IP Filtrado Decoy
- TCP Filtrado Portsweep
- UDP Filtrado Portsweep
- IP Filtrado Portsweep
- TCP Filtrado Distribuido
- UDP Filtrado Distribuido
- IP Filtrado Distribuido

5.3.1. Configuración de sfPortscan

Para poder activar el preprocesador sfPortscan, es necesario configurar y activar previamente el preprocesador *stream5*. Este preprocesador permite a sfPortscan monitorizar protocolos no orientados a conexión, como UDP, mediante la supervisión de los siguientes streams. Para llevar a cabo las diferentes pruebas de concepto que se muestran más adelante, stream5 se ha configurado de la siguiente forma:

```
preprocessor stream5_global: max_tcp 8192, track_tcp yes, track_udp yes
preprocessor stream5_tcp: policy windows, use_static_footprint_sizes,
ports client 21 22 23 25 42 53 79 80 109 110 111 113 119 135 136 137 139 143 110 111
161 445 513 514 691 1433 1521 2100 2301 3128 3306 6665 6666 6667 6668 6669 7000
8000 8080 8180 8888 32770 32771 32772 32773 32774 32775 32776 32777 32778 32779,
ports both 443 465 563 636 989 992 993 994 995
```

Para configurar sfPortscan, se encuentran disponibles las siguientes opciones:

```
proto {<proto>} (tcp, udp, icmp, ip, all)
scan-type {<scan-type>} (portscan, portsweep, decoy-portscan,
distributed-portscan, all)
sense-level {<level>} (low, medium, high)
```

Las alertas de tipo *low* se generan tan sólo cuando se reciben respuestas erróneas desde el sistema objetivo y debido a la naturaleza de este tipo de respuestas negativas, esta configuración debería generar muy pocos falsos positivos. Sin embargo, esta configuración no dará

lugar a alertas de tipo filtrado por la ausencia de respuesta negativa. El nivel low está basado en una ventana estática de 60 segundos, tras la cual, se resetea la ventana.

La configuración de tipo *medium* permite monitorizar el número de conexiones por lo que sí generarán alertas de tipo filtrado aunque puede dar lugar a falsos positivos para *hosts* activos como NATs, proxys, DNS caches, etc. por lo que será necesario utilizar directivas de tipo *ignore* para ajustar el funcionamiento del preprocesador.

En el caso de configurarse el nivel de sensibilidad como *high*, todos los *hosts* detectados en la red serán monitorizados de forma continua mediante una ventana temporal que permitirá, de forma estadística, evaluar cada caso de análisis para cada *host*. Esta configuración, permitirá detectar escáners lentos o sigilosos gracias a la continua monitorización pero es muy sensible a los *hosts* activos de la red. En cualquier caso, será necesario un ajuste inicial y pormenorizado de la configuración parte del usuario.

Otros parámetros que permiten configurar el preprocesador sfPortscan son los siguientes:

```
watch-ip { <ip1|ip2/cidr[ [port1|port2-port3]]>
```

Define las IPs, redes y puertos en dichos *hosts* que serán supervisados. Los parámetros son direcciones IP en notacion CIDR separadas por comas. De forma opcional, se pueden especificar puertos tras la dirección IP usando un espacio y se puede definir un solo puerto o un rango de puertos indicandolo con una barra inclinada. Las IPs que no pertenecen al rango especificado son ignoradas si se utiliza esta opción.

```
ignore-scanners { <ip1|ip2/cidr[ [port1|port2-port3]]> }
```

Esta directiva permite ignorar los *hosts* especificados en el caso de que puedan ser identificados como un escáner activo y generar alguna alerta. El parámetro posee el mismo formato que en el caso de la directiva watch-ip.

```
ignore-scanned { <ip1|ip2/cidr[ [port1|port2-port3]]> }
```

En este caso, esta directiva permite no generar alertas si los *hosts* pasados por parámetro son escaneados.

```
memcap { positive integer }
```

Permite definir el máximo número de bytes de memoria a utilizar para realizar la detección de escáners de puertos y cuanto mayor sea el número introducido, mayor será el número de *hosts* que pueden ser monitorizados.

```
logfile { <file> }
```

Esta opción permitira obtener una salida de los alertas y eventos generados a un fichero de texto. En el caso de que el parámetro <file>no contenga una barra '/' al comienzo, el fichero será guardado en el directorio de configuración de Snort.

```
include-midstream
```

Esta opción permitirá incluir sesiones detectadas en un punto intermedio por stream5, por lo que se pueden generar falsas alertas, especialmente cuando existe una carga alta de la red y algunos paquetes son descartados. Por ello, la opción se encuentra desactivada por defecto.

```
detect-ack-scans
```

Esta opción permitirá también supervisar sesiones detectadas en un punto intermedio por *stream5*, lo cual es necesario para detectar análisis de tipo ACK. Al igual que con el caso anterior, puede producir errores en la detección bajo condiciones de carga alta de la red.

Un ejemplo de configuración, que además se ha utilizado para realizar las pruebas de concepto que se muestran más adelante, es la siguiente:

```
preprocessor sfportscan: proto { all }
scan_type { all }
memcap { 10000000 }
sense_level { medium }
```

El aspecto más importante al configurar sfPortscan es ajustar el motor de detección de acuerdo con la arquitectura de red que se va a monitorizar. El manual de configuración nos indica algunos de los pasos a seguir [12]:

1. Usar las directivas watch-ip, ignore-scanners e ignore-scanned para conseguir un menor número de falsos positivos, adecuando la configuración al comportamiento de los *hosts* de la red

2. Las alertas marcadas como *filtered* tienen tendencia a representar falsos positivos, por lo que es importante prestar atención al tipo de alertas que se han generado a la hora de identificarlos.

3. Hacer uso de todos los elementos que componen una alerta a la hora de identificar un falso positivo:

   - **Connection Count/IP Count**: Es un ratio de conexiones por IP. Para los escáners de puertos el ratio debería ser alto mientras que para los portsweeps debería ser bajo.
   - **Port Count/IP Count**: Es un ratio de puertos conectados por IP. Debería ser alto para escáners de puertos y bajo para portsweeps.
   - **Connection Count/Port Count**: Es un ratio de conexiones por puerto. Debería ser bajo para escáners de puertos y alto para porsweeps.

4. Si todo lo demás continúa produciendo falsos positivos, podría ser conveniente disminuir el nivel de sensibilidad.

La tabla 5.3.1 es una lista de las posibles alertas que pueden ser generadas por sfPortscan y sus respectivos identificadores tal y como se muestran en la salida estándar de Snort.

5.4. NMap

Nmap es una herramienta open source diseñada para realizar tareas de exploración de la red y auditorías de seguridad, y permite identificar equipos o servicios disponibles en los equipos tanto en redes de gran tamaño como en sistemas individuales. Nmap es capaz de detectar nombres y versiones de las aplicaciones, sistemas operativos, si el tráfico está siendo filtrado por un firewall y muchos otros datos adicionales que permiten perfilar la arquitectura de una red. Por todo ello, es una herramienta estándar en las auditorías de seguridad de los sistemas y una de las preferidas por los administradores de red para controlar el estado de las redes.

La función principal de Nmap y por la cual se ha utilizado para llevar a cabo pruebas de concepto en este proyecto, es la identificación de puertos y su correspondiente estado en un sistema objetivo mediante técnicas de escáner. La salida más general de Nmap es una lista de

SID	Descripción
1	TCP Portscan
2	TCP Decoy Portscan
3	TCP Portsweep
4	TCP Distributed Portscan
5	TCP Filtered Portscan
6	TCP Filtered Decoy Portscan
7	TCP Filtered Portsweep
8	TCP Filtered Distributed Portscan
9	IP Protocol Scan
10	IP Decoy Protocol Scan
11	IP Protocol Sweep
12	IP Distributed Protocol Scan
13	IP Filtered Protocol Scan
14	IP Filtered Decoy Protocol Scan
15	IP Filtered Protocol Sweep
16	IP Filtered Distributed Protocol Scan
17	UDP Portscan
18	UDP Decoy Portscan
19	UDP Portsweep
20	UDP Distributed Portscan
21	UDP Filtered Portscan
22	UDP Filtered Decoy Portscan
23	UDP Filtered Portsweep
24	UDP Filtered Distributed Portscan
25	ICMP Sweep
26	ICMP Filtered Sweep
27	Open Port

Tabla 5.1: Lista de identificadores y alertas generadas por el preprocesador sfPortscan.

puertos donde se indica el número de puerto y protocolo, el nombre del servicio y su estado, el cual puede ser *open*, *filtered*, *closed* o *unfiltered*.

Cuando un puerto es identificado como abierto, significa que la aplicación que se encuentra escuchando en dicho puerto está a la espera y acepta conexiones. Si el puerto es identificado como filtrado, se debe a que un firewall está bloqueando las respuestas o el acceso a ese puerto y, de este modo, Nmap no puede identificar el estado real del puerto. Aunque no se conozca el estado del puerto, a efectos prácticos y desde el punto de vista de un atacante, se comporta como un puerto cerrado ya que no es posible establecer una conexión u obtener una respuesta de un ataque exitoso.

Además de la lista de puertos y su estado, Nmap puede proporcionar información adicional acerca de los objetivos, incluyendo el nombre de DNS según la resolución inversa de la IP, un listado de sistemas operativos posibles, los tipos de dispositivo, y direcciones MAC [13].

Tipos de Escáners

Nmap es capaz de realizar prácticamente todos los tipos de análisis conocidos. Algunos de ellos fueron, de hecho, creados por el desarrollador de Nmap, Fyodor, e includos en este software antes que en ningún otro. Los diferentes tipos de ataques pueden ser todos clasificados en las siguientes categorías:

- **TCP connect scan**: Este tipo de escáner conecta con el objetivo y completa un *three-way handshake* completo. Es fácilmente detectado por el sistema objetivo.
- **TCP SYN scan**: Esta técnica se conoce como *half-open scanning* porque no se realiza una conexión TCP completa. En lugar de esto, se envia un paquete SYN. Si la respuesta es un paquete SYN/TCP, el puerto está abierto, mientras que si es un RST/ACK, se encuentra cerrado. Es una técnica sigilosa pero bien soportada por los sistemas de detección de intrusión. Como se ha comentado anteriormente, Snort rastrea las respuestas negativas.
- **TCP FIN scan**: Esta técnica envía un paquete FIN al puerto objetivo, que debería responder con un RST si éste se encuentra cerrado. Es un tipo de escáner que sólo funcionará en sistemas que implementen una pila TCP/IP basada en UNIX.
- **TCP Xmas Tree scan**: Esta técnica envía un paquete con los flags FIN, URG y PUSH

activos al puerto destino. El sistema objetivo debería responder con un RST desde todos los puertos cerrados.

- **TCP Null scan**; Esta técnica desactiva todos los flags del paquete y el sistema objetivo debería responder con un RST desde todos los puertos cerrados.
- **TCP ACK scan**: Esta técnica es utilizada para detectar los conjuntos de reglas que posee un firewall. Permite determinar si el firewall tan sólo realiza filtrado de paquetes por conexión o por el contrario supervisa el estado de las conexiones realizando un filtrado de paquetes avanzado.
- **TCP Windows scan**: Esta técnica permite detectar tanto puertos abiertos como filtrados o no filtrados en algunos sistemas debido a una anomalía en el modo en que es tratado el tamaño de ventana TCP.
- **TCP RPC scan**: Es un tipo de escáner específico para sistemas UNIX y permite detectar e identificar puertos con servicios de tipo RPC (Remote Procedure Call), así como el programa asociado y su versión.
- **UDP scan**: Esta técnica envía un paquete UDP a cada puerto objetivo. Si el sistema responde con un *ICMP port unreacheable*, el puerto está cerrado, por el contrario si no se recibe este tipo de mensaje, el puerto se encuentra abierto. Debido a que UDP es un protocolo no orientado a conexión, la precisión de esta técnica depende en gran medida del filtrado y la configuración de la red objetivo.

Algunas implementaciones de TCP/IP en determinados sistemas se caracterizan por responder con paquetes que tienen el bit RST activo independientemente de como se encuentre el puerto objetivo, por lo que el resultado puede variar según el tipo de escáner que se utilice. En casos como estos o similares, las técnicas *TCP connect* y *TCP SYN* deberían funcionar en todos los sistemas.

5.4.1. Posibles tecnicas de evasion con NMAP

En algún momento durante el desarrollo de Nmap, los usuarios llegaron a sugerir que éste no debería incluir técnicas para evadir firewalls o atravesar IDSs sin ser detectado, ya que estas podrían ser utilizadas tanto por administradores de red como por atacantes. Sin

embargo, estos métodos pueden ser usados de forma maliciosa mediante otras herramientas o aplicando parches al software para añadir funcionalidades. De este modo, Nmap implementa una serie de medidas para comprobar que las configuración de los sistemas es la correcta. Éstas intentar evitar que el ataque sea detectado y conseguir saltar las restricciones impuestas por los firewalls. En este capítulo se ha intentando verificar la respuesta de Snort mediante varias pruebas de concepto en las cuales se comprueba la respuesta del IDS ante este tipo de técnicas disponibles en Nmap por defecto:

```
-f (fragmentar paquetes); --mtu (usando la MTU especificada)
```

La opción -f permite a todos los paquetes enviados por el escáner, independientemente del tipo, ser fragmentados en paquetes más pequeños. Aunque la idea es dividir la cabecera TCP entre varios paquetes para evitar ser detectado, como se ha comentado previamente, esta es una las técnicas que supervisa el preprocesador de Snort frag3. En el caso de un firewall que no realice reensamblaje, el resultado de las reglas de filtrado puede no ser el esperado por el administrador de la red. La fragmentación sólo está soportada para paquetes *raw*, lo que incluye los escáners TCP y UDP, excepto los de tipo *connect* y *FTP bounce*.

```
-D <decoy1>[,<decoy2>][,ME][,...]
```

Permite realizar un escáner de tipo decoy, de modo que el *host* remoto detectará también un escáner de los *hosts* indicados como parámetros. En este caso, el IDS detectará escáners de puertos de todas las direcciones y no podrá identificar que IP está haciendo el escáner realmente y cuales son decoys. Aunque se puede combatir esta técnica mediante rastreo de las rutas de los routers, descartando respuestas de forma selectiva y otros mecanismos activos, es, generalmente, una forma efectiva de ocultar la dirección IP del atacante. Es importante que los *hosts* usados como decoys se encuentren activos ya que sería muy sencillo determinar la IP del atacante si ésta es la única activa.

```
-S <IP-Address>
```

En algunos casos, Nmap no es capaz de determinar la dirección de origen, de modo que mostrará un mensaje de error y mediante este comando será posible especificar la IP de la interfaz a través de la cual se quieren enviar los paquetes. Otro posible uso de este flag es

espoofear el escáner para tratar de confundir al sistema objetivo o al IDS que entenderá que está siendo escaneado por otra máquina diferente a la del atacante. Sin embargo, en este caso, no se obtendrán respuestas en Nmap, ya que las respuestas serán enviadas al *host espoofeado* y no se generarán informes útiles.

`--source-port <portnumber>; -g <portnumber>`

Es muy común encontrar sistemas incorrectamente configurados que permiten transmitir o recibir tráfico según el puerto, por ejemplo, desde los puertos 53 o 20 al suponer que siempre serán respuestas DNS o FTP respectivamente. Nmap ofrece las opciones -g y –source-port para explotar estas vulnerabilidades, realizando los ataques de reconocimiento desde los puertos especificados. La mayoría de análisis TCP, incluido el de tipo SYN, y de tipo UDP, soportan esta opción.

`--data-length <number>`

Nmap utiliza habitualmente paquetes pequeños que contienen tan sólo la cabecera TCP, de modo que los paquetes enviados tienen 40 bytes y las respuestas ICMP 28. Esta opción permite añadir el número especificado de bytes aleatorios a la mayoría de paquetes enviados. La mayoría de pings y escáners de puertos soportan esta opción y a pesar de que puede ralentizar el escáner, también permite hacerlo más sigiloso.

`--ip-options <S|R [route]|L [route]|T|U ... >; --ip-options <hex string>`

El protocolo IP ofrece varias opciones que pueden definirse en las cabeceras. Al contrario que las opciones TCP, las opciones IP rara vez se establecen por cuestiones de funcionalidad y seguridad. Muchos routers en internet bloquean las opciones mas peligrosas como el *source routing*, aunque algunas opciones pueden ser útiles para determinar y manipular la ruta hasta el objetivo a través de la red.

`--ttl <value>`

Permite fijar el campo *time-to-live* al valor fijado en los paquetes enviados.

`--randomize-hosts`

Le indica a Nmap que debe mezclar de forma aleatoria el orden de cada grupo de hasta 16384 *hosts* antes de escanearlos. Esto puede hacer el escáner menos evidente ante varios sistemas que estén monitorizando la red, especialmente cuando es combinado con una temporización lenta.

`--spoof-mac <MAC address, prefix, or vendor name>`

Permite especificar la dirección MAC de todos los paquetes *raw* ethernet que se envíen. Esta opción sólo afecta a los paquetes de un escáner SYN o de detección del sistema operativo, no a los análisis orientados a conexión o a la detección de versiones de servicios.

`--badsum`

Le indica a Nmap que debe usar un checksum erróneo para paquetes TCP, UDP y SCTP. Como la mayoría de implementaciones IP en los *hosts* descartan estos paquetes, cualquier respuesta que se reciba habrá sido enviada por un firewall o un IDS que no ha comprobado el checksum.

`--min-rate <number>; --max-rate <number>`

Permite especificar la tasa de envío de paquetes. Estas opciones están diseñadas para situaciones en las que la tasa de envío está limitada por las características de la red y un *host* no puede escanearse demasiado rápido o para un caso en el que el escáner deba completarse en un tiempo determinado. Ambas opciones son globales y afectan a todos los paquetes enviados para un escáner, independientemente del *host* objetivo al que se envíen. Una técnica muy efectiva para evitar la detección del IDS usando esta opción, es especificar un tasa de envío muy baja, que permita confundir a los temporizadores presentes en los motores de detección.

`-T paranoid | sneaky | polite | normal | aggressive | insane`

Esta opción, al igual que la opción anterior, permite especificar una tasa de envío de paquetes de forma más general y utilizando varias opciones estándar. Los controles más específicos de la opción anterior son muy efectivos, pero la selección, en algunos casos, de la tasa correcta, puede llevar más tiempo que el mismo escáner. Por ello, Nmap ofrece seis plantillas de temporización que pueden ser indicadas con el flag -T y un número de 0 a 5. Estas plantillas, permiten al usuario especificar que nivel de agresividad puede tener el escáner, dejando

a Nmap la elección exacta de la tasa de envío. Las posibles opciones son *paranoid* (0), *sneaky* (1), *polite* (2), *normal* (3), *aggressive* (4), e *insane* (5). Las dos primeras están ideadas para conseguir evadir un IDS, el modo polite es algo mas lento de lo normal y permite consumir menor ancho de banda y recursos de la máquina objetivo, el modo normal es el que se activa por defecto, el modo agressive acelera el envío asumiendo que la red es razonablemente rápida y, finalmente, el modo aggressive, que sólo es apto para redes muy rápidas y en casos en los que puede sacrificarse precisión en los resultados por la velocidad del escáner. Aunque las opciones -T0 y -T1 son útiles para evitar generar alertas en un IDS, llevarán un tiempo considerablemente largo para completarse para redes grandes. El principal efecto de T0 es serializar el escáner de modo que se escanea por completo un puerto cada vez y se esperan cinco minutos para realizar cada envío de paquetes. T1 y T2 son similares pero tan sólo esperan 15 y 0,4 segundos. T3 es el comportamiento por defecto de Nmap e incluye paralelización. T4 es un equivalente de *–max-rtt-timeout 1250 –initial-rtt-timeout 500 –max-retries 6* y fija el retardo máximo del escáner TCP a 10 milisegundos. T5 equivale a *–max-rtt-timeout 300 –min-rtt-timeout 50 –initial-rtt-timeout 250 –max-retries 2 –host-timeout 15m* y fija el máximo retardo en análisis TCP a 5 milisegundos.

5.5. Pruebas de Concepto

En este apartado se analizan los resultados de una serie de pruebas de concepto realizadas a partir de algunas de las opciones de las que dispone Nmap y que han sido presentadas en el apartado anterior. Con ellas, se pretende comprobar la respuesta del IDS Snort ante análisis con paquetes modificados o que incluyen algún tipo de técnica cuyo objetivo es evadir o confundir al sistema de detección.

5.5.1. Escáner UDP con decoy del router de la red desde el puerto 67

En esta prueba de concepto, el sistema 192.168.0.146 realiza un análisis sobre el sistema 192.168.0.199 con paquetes UDP y con la opción *decoy* activa. Se ha seleccionado la IP del dispositivo router y puerta de enlace de la red y el puerto 67, utilizado por el mismo para transmitir las respuestas del servidor DHCP. La idea de esta prueba es comprobar si el IDS detecta el origen real del escáner y lo identifica a pesar de ser paquetes que podrían ser enviados por el servidor DHCP auténtico desde el router de la red.

```
hugo@HG01:~$ sudo nmap -T3 -D 192.168.0.1 -sU --source_port 67 -PN 192.168.0.199

Starting Nmap 4.76 ( http://nmap.org ) at 2009-07-30 21:05 CEST
Interesting ports on 192.168.0.199:
Not shown: 996 closed ports
PORT     STATE         SERVICE
68/udp   open|filtered dhcpc
137/udp  open|filtered netbios-ns
138/udp  open|filtered netbios-dgm
3130/udp open|filtered squid-ipc
MAC Address: 00:23:12:57:C9:9E (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1075.39 seconds
hugo@HG01:~$
```

Figura 5.2: Análisis con NMap en la PoC #1.

La figura 5.2 muestra el resultado del escáner en la salida de la consola. Se comprueba que aunque es capaz de identificar algunos puertos como abiertos, no todos los puertos abiertos son detectados y los que se detectan son marcados como filtrados.

La figura 5.3 muestra las alertas generadas por Snort y recogidas por la interfaz BASE. Como se puede ver, Snort ha detectado el origen del escáner en la máquina 192.168.0.146 y alerta de un intento de ataque desde el puerto 67 de ambas máquinas al puerto 162. Lo mismo ocurre para las peticiones realizadas al puerto 5060, donde se recibe una respuesta VOIP-SIP demasiado pequeña.

5.5.2. Escáner TCP SYN con fragmentación de paquetes

En esta prueba de concepto, el sistema 192.168.0.146 realiza un análisis sobre el sistema objetivo con peticiones TCP SYN transmitidas en paquetes fragmentados. Intentamos comprobar la capacidad de Snort para rastrear las respuesta negativas de la máquina objetivo y reensamblar las distintas tramas recibidas.

La figura 5.4 muestra el resultado del escáner en la salida de la consola. Se comprueba que en este caso se identifican correctamente los servicios disponibles en la máquina objetivo y el tiempo del escáner es considerablemente más rápido que en el caso anterior. El estado de los puertos, marcados como abiertos, es identificado correctamente.

La figura 5.5 muestra las alertas generadas por Snort y recogidas por la interfaz BASE. Observamos que el escáner es detectado por Snort correctamente, así como las respuestas

ID	< Signature >	< Timestamp >	< Source Address >	< Dest. Address >	< Layer 4 Proto >
#0-(1-101)	[snort] (portscan) UDP Portscan: 402:57409	2009-07-25 11:14:41	192.168.0.146	192.168.0.199	Raw IP
#1-(1-100)	[snort] (portscan) UDP Portscan: 502:53838	2009-07-25 11:13:39	192.168.0.146	192.168.0.199	Raw IP
#2-(1-99)	[snort] (portscan) UDP Portscan: 683:55544	2009-07-25 11:12:37	192.168.0.146	192.168.0.199	Raw IP
#3-(1-96)	[snort] (portscan) UDP Portscan: 17:49393	2009-07-25 11:11:37	192.168.0.146	192.168.0.199	Raw IP
#4-(1-95)	[snort] (portscan) UDP Portscan: 631:49226	2009-07-25 11:10:37	192.168.0.146	192.168.0.199	Raw IP
#5-(1-98)	[cve] [icat] [cve] [icat] [bugtraq] [bugtraq] [bugtraq] [local] [snort] SNMP trap udp	2009-07-25 11:12:15	192.168.0.1:67	192.168.0.199:162	UDP
#6-(1-97)	[cve] [icat] [cve] [icat] [bugtraq] [bugtraq] [bugtraq] [local] [snort] SNMP trap udp	2009-07-25 11:12:15	192.168.0.146:67	192.168.0.199:162	UDP
#7-(1-94)	[snort] (portscan) UDP Portscan: 18081:49159	2009-07-25 11:09:35	192.168.0.146	192.168.0.199	Raw IP
#8-(1-91)	[snort] (portscan) UDP Portscan: 6004:21948	2009-07-25 11:08:35	192.168.0.146	192.168.0.199	Raw IP
#9-(1-90)	[snort] (portscan) UDP Portscan: 983:40441	2009-07-25 11:07:33	192.168.0.146	192.168.0.199	Raw IP
#10-(1-93)	[url] [local] [snort] VOIP-SIP response too small	2009-07-25 11:09:06	192.168.0.1:67	192.168.0.199:5060	UDP
#11-(1-92)	[url] [local] [snort] VOIP-SIP response too small	2009-07-25 11:09:06	192.168.0.146:67	192.168.0.199:5060	UDP
#12-(1-89)	[snort] (portscan) UDP Portscan: 68:58797	2009-07-25 11:06:36	192.168.0.146	192.168.0.199	Raw IP
#13-(1-88)	[snort] (portscan) UDP Portscan: 434:39217	2009-07-25 11:05:31	192.168.0.146	192.168.0.199	Raw IP
#14-(1-87)	[snort] (portscan) UDP Portscan: 514:61961	2009-07-25 11:04:30	192.168.0.146	192.168.0.199	Raw IP
#15-(1-86)	[snort] (portscan) UDP Portscan: 16912:49167	2009-07-25 11:03:29	192.168.0.146	192.168.0.199	Raw IP
#16-(1-80)	[snort] (portscan) UDP Portscan: 3:63555	2009-07-25 10:57:19	192.168.0.146	192.168.0.199	Raw IP
#17-(1-81)	[snort] (portscan) UDP Portscan: 20:57977	2009-07-25 10:58:24	192.168.0.146	192.168.0.199	Raw IP
#18-(1-82)	[snort] (portscan) UDP Portscan: 1088:40711	2009-07-25 10:59:26	192.168.0.146	192.168.0.199	Raw IP
#19-(1-83)	[snort] (portscan) UDP Portscan: 13:62154	2009-07-25 11:00:25	192.168.0.146	192.168.0.199	Raw IP
#20-(1-84)	[snort] (portscan) UDP Portscan: 37:51690	2009-07-25 11:01:26	192.168.0.146	192.168.0.199	Raw IP
#21-(1-85)	[snort] (portscan) UDP Portscan: 18449:54711	2009-07-25 11:02:28	192.168.0.146	192.168.0.199	Raw IP

Figura 5.3: Alertas generadas en BASE en la PoC #1.

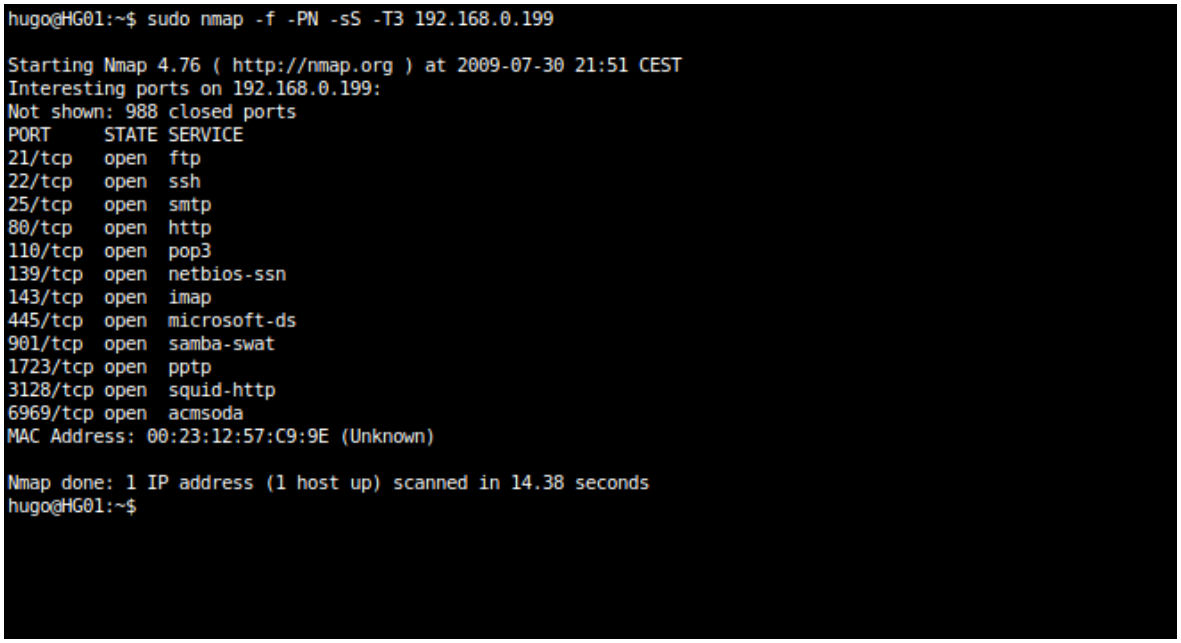


Figura 5.4: Análisis con NMap en la PoC #2.

positivas de los puertos abiertos. Además, se observan otro tipo de alertas: se detectan pings ICMP cuando se ha pasado específicamente a Nmap la opción de no realizar ningún ping y se generan alertas para vulnerabilidades dirigidas a los puertos 705 y 161. Estos tres tipos de alertas pueden considerarse falsos positivos.

5.5.3. Escáner TCP *connect* con fragmentación de paquetes

En esta prueba de concepto, el sistema 192.168.0.146 realiza un análisis sobre el sistema objetivo con peticiones TCP connect y paquetes fragmentados. Si bien este tipo de análisis es menos sigiloso que el de tipo SYN, puede ser identificado como una conexión legítima y producirse un falso negativo.

La figura 5.6 muestra el resultado del escáner en la salida de la consola. Se comprueba que en este caso los servicios disponibles también son identificados por el escáner correctamente así como su estado. El tiempo de ejecución es similar al de un escáner de tipo SYN.

La figura 5.7 muestra las alertas generadas por Snort y recogidas por la interfaz BASE. También en este caso, se observa como Snort es capaz de detectar el escáner así como las respuestas de los puertos abiertos. En comparación con el caso anterior, no se han detectado

Queried on : Sat July 25, 2009 11:43:51

Meta Criteria	*any*
IP Criteria	*any*
Layer 4 Criteria	*none*
Payload Criteria	*any*

Summary Statistics

- **Sensors**
- **Unique Alerts**
- (**classifications**)
- Unique addresses: **Source** | **Destination**
- **Unique IP links**
- **Source** Port: **TCP** | **UDP**
- **Destination** Port: **TCP** | **UDP**
- **Time profile** of alerts

Displaying alerts 1-14 of 14 total

ID	< Signature >	< Timestamp >	< Source Address >	< Dest. Address >	< Layer 4 Proto >
#0-(1-102)	[local] [snort] ICMP PING *NIX	2009-07-25 11:43:24	192.168.0.146	192.168.0.199	ICMP
#1-(1-103)	[local] [snort] ICMP PING	2009-07-25 11:43:24	192.168.0.146	192.168.0.199	ICMP
#2-(1-104)	[local] [snort] ICMP PING *NIX	2009-07-25 11:43:25	192.168.0.146	192.168.0.199	ICMP
#3-(1-105)	[local] [snort] ICMP PING	2009-07-25 11:43:25	192.168.0.146	192.168.0.199	ICMP
#4-(1-106)	[snort] (portscan) TCP Portscan: 21:3389	2009-07-25 11:43:37	192.168.0.146	192.168.0.199	Raw IP
#5-(1-107)	[snort] (portscan) Open Port: 21	2009-07-25 11:43:37	192.168.0.146	192.168.0.199	Raw IP
#6-(1-108)	[snort] (portscan) Open Port: 25	2009-07-25 11:43:37	192.168.0.146	192.168.0.199	Raw IP
#7-(1-109)	[snort] (portscan) Open Port: 1723	2009-07-25 11:43:37	192.168.0.146	192.168.0.199	Raw IP
#8-(1-110)	[snort] (portscan) Open Port: 80	2009-07-25 11:43:37	192.168.0.146	192.168.0.199	Raw IP
#9-(1-111)	[snort] (portscan) Open Port: 22	2009-07-25 11:43:38	192.168.0.146	192.168.0.199	Raw IP
#10-(1-114)	[cve] [icat] [cve] [icat] [bugtraq] [bugtraq] [bugtraq] [local] [snort] SNMP AgentX/tcp request	2009-07-25 11:43:39	192.168.0.146:47462	192.168.0.199:705	TCP
#11-(1-115)	[cve] [icat] [cve] [icat] [bugtraq] [bugtraq] [bugtraq] [local] [snort] SNMP request tcp	2009-07-25 11:43:40	192.168.0.146:47462	192.168.0.199:161	TCP
#12-(1-112)	[snort] (portscan) Open Port: 445	2009-07-25 11:43:39	192.168.0.146	192.168.0.199	Raw IP
#13-(1-113)	[snort] (portscan) Open Port: 901	2009-07-25 11:43:39	192.168.0.146	192.168.0.199	Raw IP

Figura 5.5: Alertas generadas en BASE en la PoC #2.

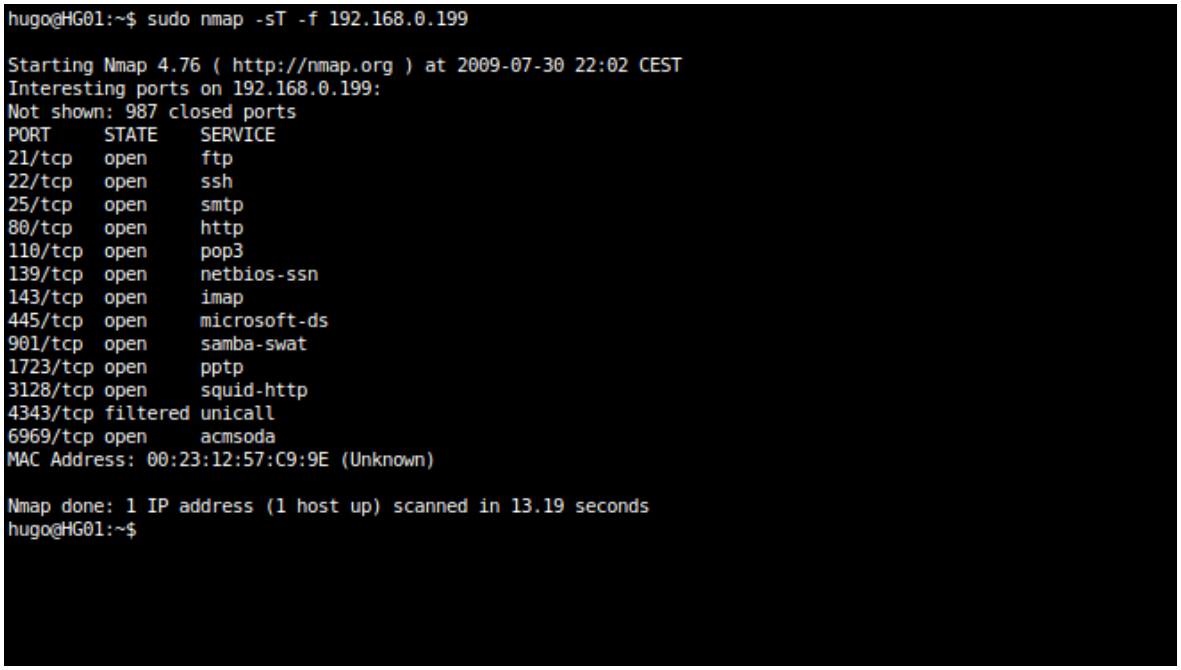


Figura 5.6: Análisis con NMap en la PoC #3.

pings ICMP aunque continúan generándose alertas para ataques a los puertos 705 y 161. Esto indica que las peticiones TCP SYN de Nmap son analizadas por el motor de reglas de Snort y al no coincidir con la estructura SNMP correcta de la petición de servicio asociada a dicho puerto, son marcadas como ataques, constituyendo falsos positivos.

5.5.4. Escáner TCP SYN con checksum de paquetes incorrecto

En esta prueba de concepto, el sistema 192.168.0.146 realiza un análisis sobre el sistema objetivo con peticiones TCP SYN que poseen un checksum erróneo. Parece evidente que la implementación TCP/IP de la máquina 192.168.0.199, independientemente del sistema operativo activo, descartará estas peticiones por ser su checksum incorrecto. El objetivo de este test es comprobar de que forma el modo de sensibilidad *medium* del preprocesador sfPortscan, permite identificar el escáner en ausencia de respuestas negativas del objetivo. El nivel *low* es el que se encuentra definido por defecto en el archivo de configuración *snort.conf* para el sfPortscan, pero como se ha explicado en apartados anteriores, este nivel de sensibilidad no permite rastrear un escáner en ausencia de respuestas negativas.

La figura 5.8 muestra el resultado del escáner en la salida de la consola, mientras que

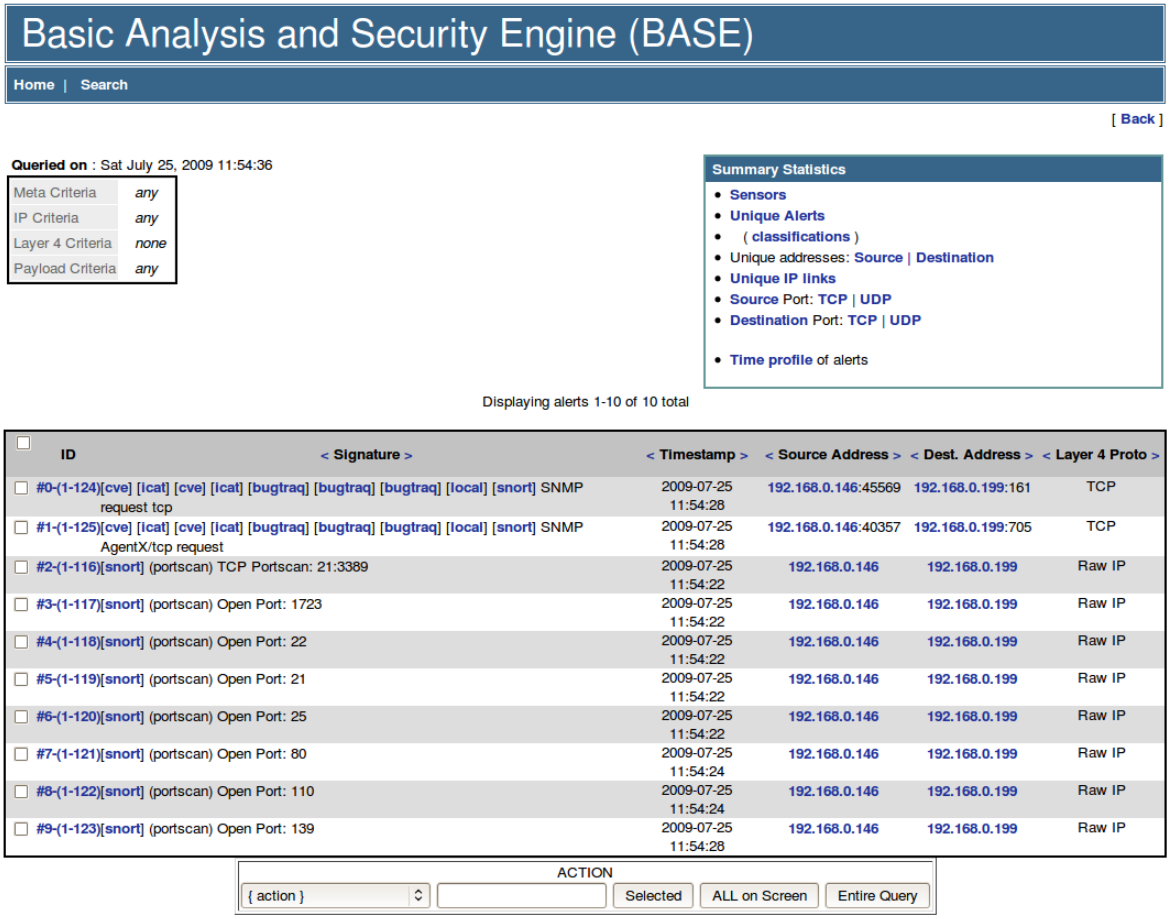


Figura 5.7: Alertas generadas en BASE en la PoC #3.

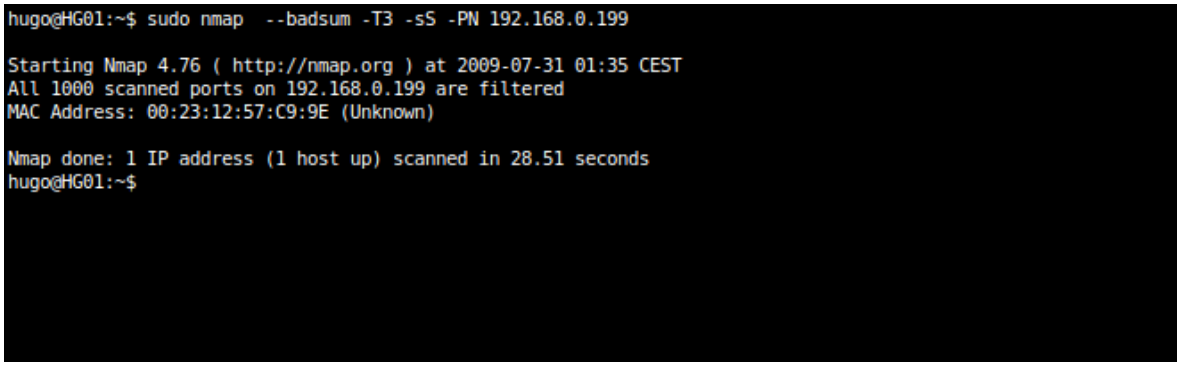


Figura 5.8: Análisis con NMap en la PoC #4.

la figura 5.9 representa el diagrama de flujo de parte del escáner obtenido con el programa *Wireshark*. Se observa como, efectivamente, la máquina 192.168.0.199 descarta todas las peticiones y no transmite ninguna respuesta afirmativa o negativa. El escáner muestra como resultado que todos los puertos del sistema se encuentran filtrados, por lo que no es posible conocer su estado independientemente del mismo. El tiempo total del escáner es ligeramente mayor que en los dos casos anteriores como consecuencia de las esperas asociadas a los timers de TCP.

La figura 5.10 muestra las alertas generadas por Snort y recogidas por la interfaz BASE. Se observa como Snort detecta el escáner aunque de forma incorrecta. A pesar de que todas las peticiones son de tipo TCP, Snort detecta que un 50 % de los paquetes se corresponden con peticiones UDP. Además, todas las peticiones son identificadas como un escáner de tipo portsweep, esto es, de un único atacante a varios objetivos distintos, cuando en realidad, todas las peticiones están dirigidas a un único objetivo. Se demuestra, por tanto, que el análisis de tramas con un checksum erróneo también afecta a una correcta identificación de los mismos por parte de Snort.

5.5.5. Escáner TCP SYN en modo *stealth* (-T0)

En esta prueba de concepto, el sistema 192.168.0.146 realiza un análisis sobre el sistema objetivo con peticiones TCP SYN enviadas con un intervalo de tiempo de cinco minutos entre las mismas. El objetivo de esta prueba es comprobar si Snort es capaz rastrear ataques de reconocimiento sigilosos realizados con una tasa de envío muy baja.

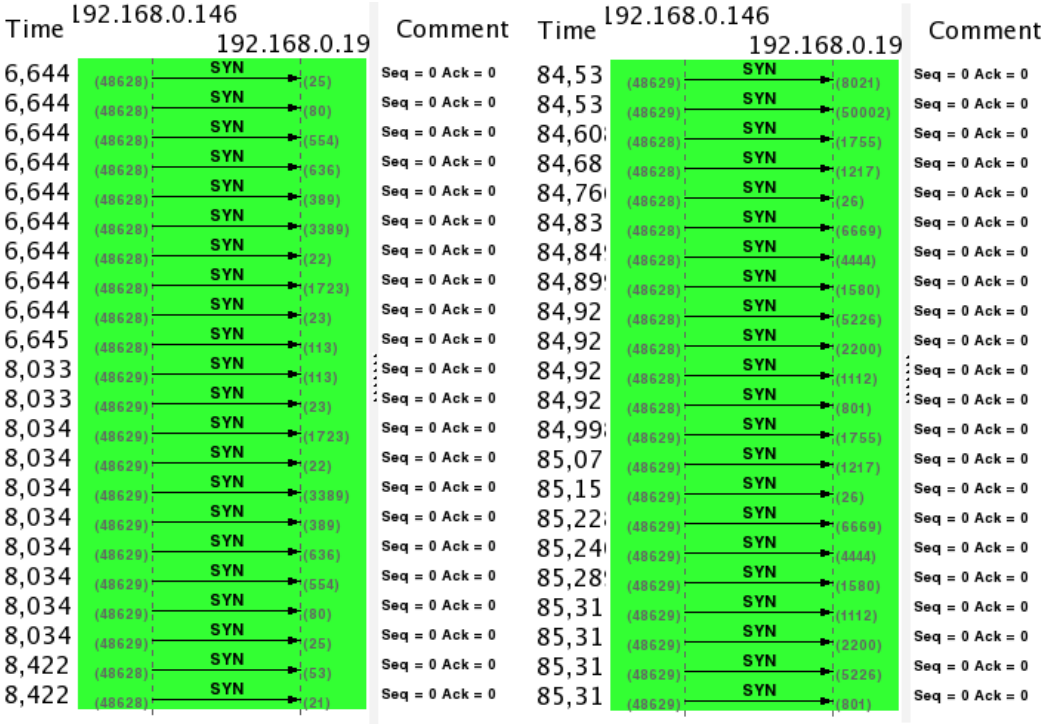


Figura 5.9: Inicio y fin del flujo de tráfico del escáner en la PoC #4 (Wireshark).

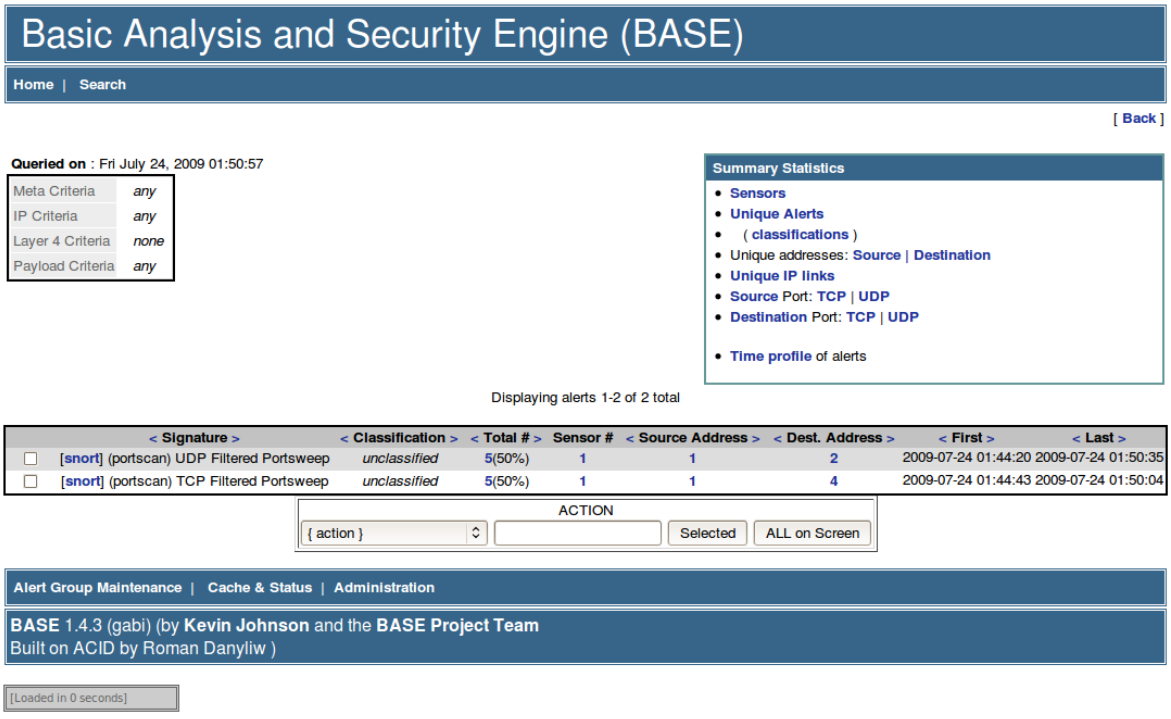


Figura 5.10: Alertas generadas en BASE en la PoC #4. sfPortscan en level medium

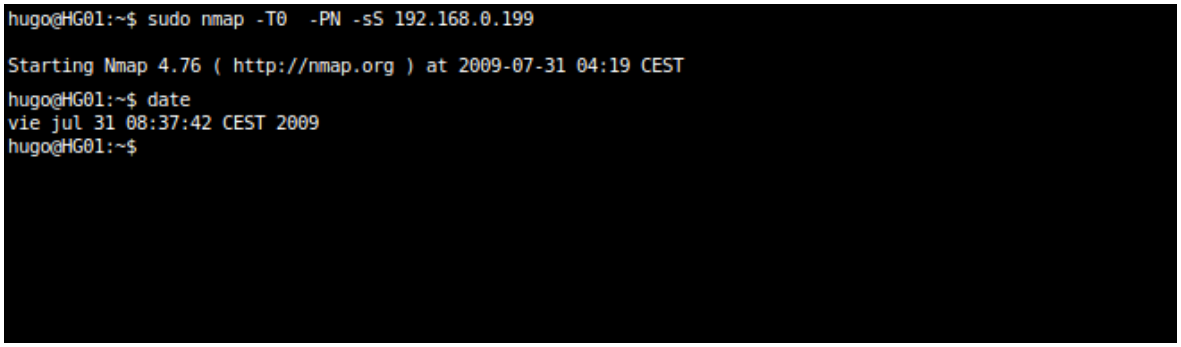


Figura 5.11: Análisis con NMap en la PoC #5.

La figura 5.11 muestra el resultado del escáner en la salida de la consola, mientras que la figura 5.12 representa la última parte del perfil temporal de los paquetes transmitidos y recibidos durante el escáner obtenido a partir del sniffer Wireshark. El tiempo total durante el que se ha llevado a cabo el escáner ha sido de 4 horas, 15 minutos y 34 segundos. En la figura se puede comprobar la regularidad, cada cinco minutos, de los intervalos de tiempo en la transmisión de las peticiones por parte del escáner (negro) y la respuesta positiva o negativa del objetivo (rojo). La unidad de longitud de las barras en el eje Y representa el número de bytes transmitidos o recibidos, 50 para las peticiones del escáner y 60 para las respuestas, de ahí su mayor longitud. En esta gráfica se pretende mostrar la regularidad de las peticiones así como la amplia duración del escáner en el tiempo, de modo que las peticiones y las respuestas aparecen superpuestas. Si se considera un intervalo de tiempo lo suficientemente pequeño se puede observar el momento preciso en el que se realiza cada petición y en el que se recibe la respuesta unos 0,15 milisegundos en media después.

La figura muestra las alertas generadas por Snort y recogidas por la interfaz BASE. Se puede observar como a pesar de los amplios intervalos de tiempo entre peticiones el IDS Snort es capaz de detectar que se está llevando a cabo un escáner de puertos. No obstante y como ocurría en casos anteriores, se detectan algunos pings ICMP a pesar de haber pasado a Nmap la opción de no realizar pings sobre el objetivo y el 97 % de las peticiones se identifican como un escáner de tipo UDP portsweep. De este modo, se comprueba que a pesar de que la detección realizara por Snort no es del todo correcta y da lugar a falsos positivos, si es capaz de detectar la presencia de un ataque de reconocimiento en curso a pesar de la baja tasa de envío.

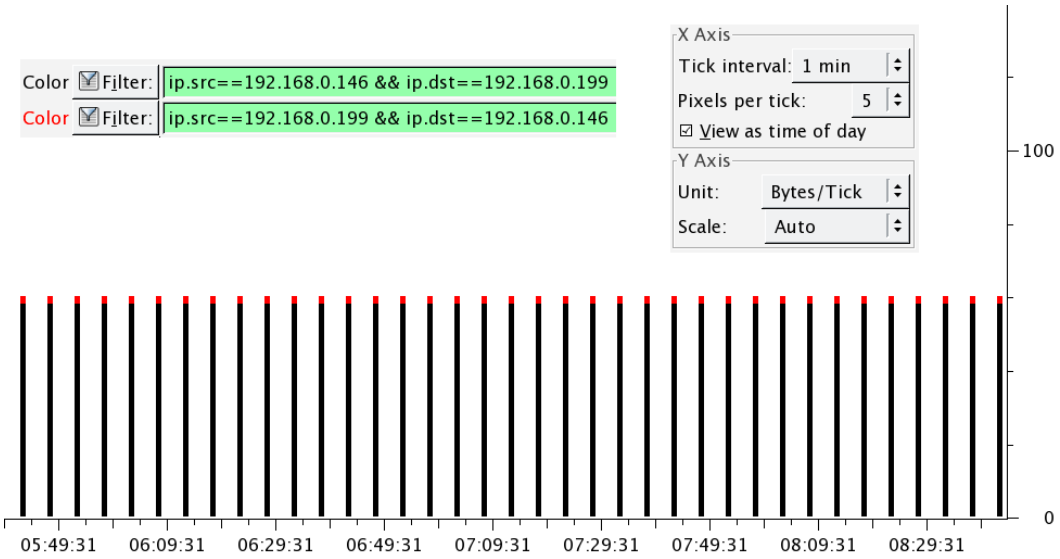


Figura 5.12: Perfil temporal de tráfico del escáner en la PoC #5 (Wireshark).

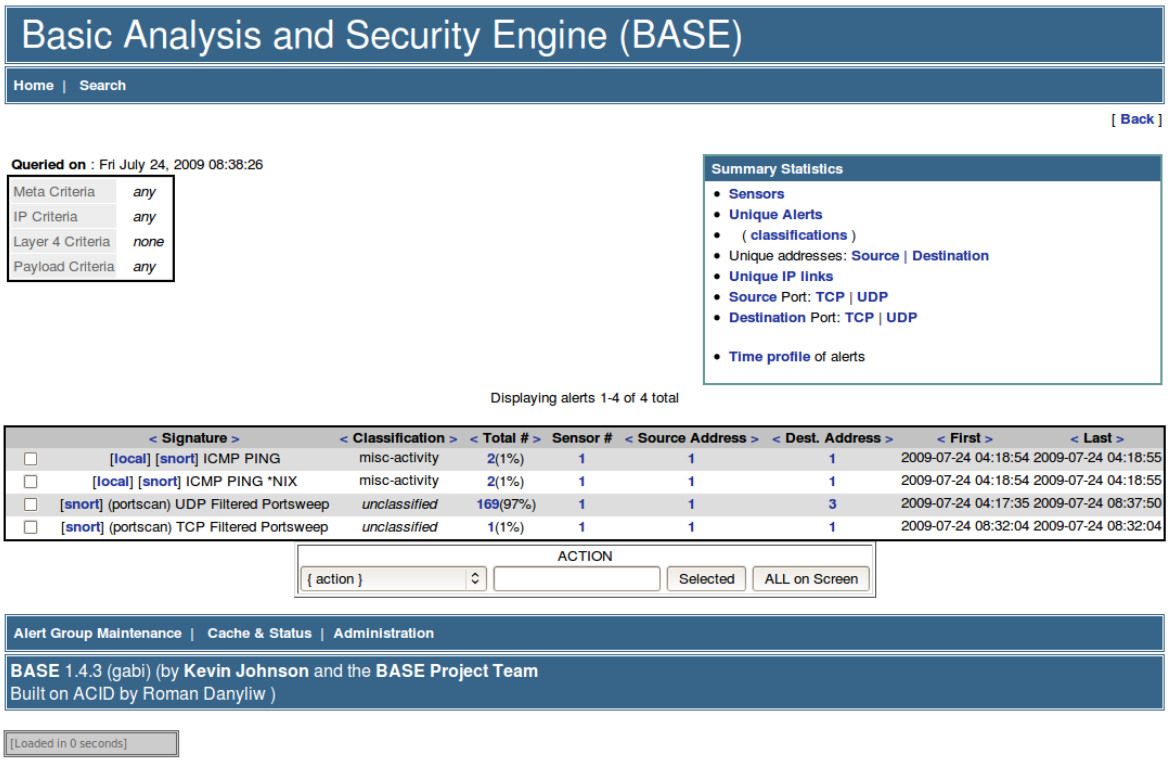


Figura 5.13: Alertas generadas en BASE en la PoC #5.

5.6. Conclusiones

A partir de las pruebas de concepto realizadas se ha comprobado la alta capacidad de Snort para detectar ataques de reconocimiento de puertos. Si bien es cierto que una evasión efectiva requiere una gran técnica y esfuerzo considerable por parte de un atacante, las pruebas realizadas con opciones estándar presentes en el escáner más utilizado permiten dar una idea aproximada de la capacidad del IDS Snort para generar alertas a partir de este tipo de ataques.

En algunos casos se ha visto cómo se pueden generar falsos positivos, lo que puede requerir un ajuste más fino de las opciones de configuración tanto de los preprocesadores como de las reglas de Snort. La mejor detección se observa para ataques de tipos muy conocidos como son los de tipo TCP SYN o TCP Connect, en los cuales un rastreo de las conexiones y las respuestas negativas es suficiente para identificar la mayoría de parámetros del escáner. Otros tipos de análisis más complejos como aquellos que intentan confundir al IDS pueden dar lugar a una detección positiva pero también a la imposibilidad de detectar al atacante, como puede ocurrir en un ataque de tipo *idle scan* [13] en el que el tan sólo pueden rastrearse las respuestas negativas y las respuestas erróneas del sistema zombie, pero nunca al verdadero atacante.

Cabe destacar la capacidad de Snort para detectar ataques sigilosos en los cuales las peticiones se realizan con un gran intervalo de tiempo entre ellas. Aunque en la prueba de concepto realizada se ha utilizado la plantilla de tiempo con la tasa más baja ofrecida por Nmap, teóricamente, sería posible realizar un ataque con la opción –max-rate <number>fijada a un intervalo de tiempo superior a los timers implementados por los preprocesadores de sfPortscan para guardar en memoria la información relativa a cada flujo. En tal caso es posible que la duración del escáner se prolongara durante días.

Capítulo 6

Detección de Vulnerabilidades

6.1. Introducción

Los sistemas de detección de intrusión están diseñados para analizar el tráfico de la red en busca de evidencias de actividad maliciosa. Cuando el algoritmo de detección del IDS genera una alerta para tráfico que no constituye una actividad maliciosa o sospechosa, se dice que se ha producido un *falso positivo*. Es importante matizar que desde las perspectiva del IDS, éste no está funcionando de manera incorrecta, es decir, su motor de detección no está cometiendo un error ya que se entiende que el algoritmo no es perfecto.

Un ejemplo puede ser la búsqueda de URLs extremadamente largas. Típicamente, una URL puede tener sólo 500 bytes y configurar el IDS para generar una alerta cuando se transmite una URLs mayor de 2000 bytes puede servir para identificar un intento de ataque basado en la denegación de servicio. Un falso positivo podría resultar de la visita a una web de venta online en la cual se puede generar una URL muy compleja y a partir de la cual el servidor almacena información de los datos contenidos en la misma. En algunos casos, este tipo de URLs pueden superar la longitud límite de 2000 bytes. Otro ejemplo podría ser la búsqueda de una referencia al fichero de Unix */etc/passwd* en una URL. Las peticiones o referencias a este fichero son típicas de ataques a servidores web que tienen activa la ejecución de archivos binarios (CGI-BIN). Un IDS que buscara simplemente la aparición de */etc/passwd* en la URL de una WEB, generaría un falso positivo cuando un usuario intenta realizar una búsqueda en Google sobre el fichero */etc/passwd* [11].

La mayoría de los sistemas de detección de intrusión generan un gran número de falsos

positivos cuando son puestos en funcionamiento con su configuración por defecto y es tarea del administrador del sistema el ir ajustando dicha configuración de acuerdo con los requisitos de la red para reducir su número. Sin embargo y por contra, aunque los falsos positivos son problemáticos desde el punto de vista de la efectividad y optimización del IDS, el hecho de que se produzcan falsos negativos puede ser mucho mas peligroso para la seguridad de la red. Se produce un falso negativo cuando el IDS no genera ninguna alerta ante un ataque real a un equipo de la red. Los falsos positivos se pueden descartar pero es imposible detectar la presencia de falsos negativos a priori.

La forma más habitual de corregir la aparición de falsos positivos es a partir del conocimiento que se tiene sobre la estructura y el funcionamiento de la red que se está defendiendo. Aunque algunas reglas detecten un tipo de ataque, es posible que la relevancia de dicho ataque en el entorno de red en cuestión sea nula porque no existe ese servicio disponible en ningún equipo. En tal caso, bastaría con desactivar dicha regla para optimizar el comportamiento del IDS, reduciendo su consumo computacional y la cantidad de 'ruido' generado. Sin embargo a la hora de comprobar la existencia de falsos negativos, no existe otra posibilidad que probar la configuración del IDS mediante ataques selectivos que permitan verificar la detección que el IDS realiza de los mismos.

Una de las técnicas más comunes a la hora de verificar la configuración de un IDS es la utilización de un sistema de detección de vulnerabilidades (*VDS, Vulnerability Detection Scanner*). Un VDS permite, típicamente, realizar una serie de ataques sistematizados sobre una red o un equipo de red para detectar la presencia de vulnerabilidades. Un IDS situado en dicho entorno de red ha de ser capaz de reconocer los ataques generados por el VDS cuando éste indique la presencia de vulnerabilidades a partir de un intercambio de paquetes con la máquina objetivo. Si el VDS es capaz de detectar una vulnerabilidad y el IDS no ha generado ninguna alerta, es posible que se esté produciendo un falso negativo y. por tanto, una posible situación de riesgo.

En el capitulo 4 se he demostrado de forma teórica, a partir de las actualizaciones de reglas para detección de Snort y los plugins para detección de vulnerabilidades del escáner Nessus, que en algunos casos, la configuración del motor de reglas del IDS puede sufrir falsos negativos cuando se han publicado *exploits* para realizar ataques sobre vulnerabilidades que Snort todavía no es capaz de detectar. El escáner Nessus, que permite verificar la configuración de Snort también puede disponer en algunos casos de plugins para los que el IDS no tiene

detección. El objetivo de este capítulo es comprender y analizar el funcionamiento del escáner de vulnerabilidades Nessus y verificar en qué medida la respuesta del IDS Snort es adecuada en relación con la presencia tanto de falsos negativos como de falsos positivos.

6.2. Nessus Vulneraribility Scanner

El escaner Nessus se puede considerar el más popular de los escaners de vulnerabilidades activos y ha sido una herramienta libre y de código abierto hasta la publicación en 2005 de su versión 3.0, la cual abandonó la licencia GPL para ser liberada, junto con una serie de modificaciones en su sistema de actualizaciones, bajo una nueva licencia propietaria.

Aunque ha sido desarrollado tradicionalmente para sistemas Unix, ya se encuentran disponibles versiones para la mayoría de los sistemas operativos. Su función principal es la de descubrir de forma rápida la existencia de vulnerabilidades conocidas en un sistema, aunque las últimos desarrollos han incluido funcionalidades que permiten realizar auditorías de configuración y descubrimiento de información sensible en los sistemas auditados [7].

6.2.1. Configuración y Funcionalidades

Nessus sigue un modelo cliente-servidor donde el segundo, *nessusd*, se ejecuta como demonio en una máquina UNIX. Desde el servidor se lleva a cabo el proceso de análisis por medio de análisis de la red o el sistema objetivo. El demonio puede ser programado para hacer análisis periódicos con la utilidad *cron*.

Varios tipos de clientes pueden acceder al servidor desde diferentes tipos de sistemas (Microsoft Windows, OS X, Zarus o via interfaz web). Existe un front-end basado en las librerias GTK para X11, también llamado *nessus*, que además de permitir una configuración mas sencilla del proceso de análisis, muestra el avance y un informe final del resultado una vez que el análisis se ha completado. El demonio *nessusd* puede también ser funcional directamente desde la consola.

Una vez que se ha completado el análisis, los datos aparecen en el cliente y pueden ser descartados o guardados en diversos formatos. Los resultados pueden ser exportados a texto plano para un posterior parseo con mayor facilidad, XML, HTML, y LaTeX. Existe además un formato propio que puede ser cargado posterioremente en el cliente o guardado en una base de conocimiento para referencia en futuros análisis de vulnerabilidades [**?**, 3].

En el modo de operación más habitual, Nessus realiza en primer lugar un escaneado de puertos de la máquina objetivo con el fin de detectar servicios disponibles y puertos abiertos. El siguiente paso es llevar a cabo un ataque sistemático contra dichos servicios para probar la existencia de vulnerabilidades. Esto se lleva a cabo mediante los diferentes exploits disponibles en su base de datos y los cuales son funcionales en forma de plugins. Los diferentes exploits son escritos en NASL.

Algunas de las pruebas de vulnerabilidades de Nessus pueden provocar una denegación de servicio al causar que los servicios o sistemas operativos fallen y se caigan, de modo que el usuario puede evitar que esto ocurra desactivando la opción *unsafe test* (pruebas no seguras) en el cliente gráfico antes de escanear.

6.2.2. El lenguaje NASL

NASL (Nessus Attack Scripting Language, Lenguaje de Scripting de Ataque Nessus por sus siglas en inglés) es un lenguaje de scripting especificamente diseñado para Nessus aunque su curva de aprendizaje es rápida, sobre todo si el usuario posee conocimientos previos de C, lenguaje con el que comparte elementos de sintaxis. Sus objetivos son permitir a cualquier usuario escribir un test para una vulnerabilidad dada de forma rápida o compartir dichos tests con independencia del sistema operativo desde el que vayan a utilizarse. De este modo, NASL permite crear paquetes de distintos protocolos especialmente formados para explotar una vulnerabilidad determinada y proporciona funciones que permiten escrbir tests para servidores web y ftp de forma sencilla.

NASL garantiza que un script no enviará ningún paquete a un *host* diferente del *host* objetivo y que no se ejecutará ningun comando en el sistema local. La claridad y simpleza del lenguaje permiten, además, garantizar a cualquier usuario que el código de un plugin escrito en NASL realizará tan sólo el test de seguridad esperado contra el sistema objetivo y no llevará a cabo ningún tipo de actividad maliciosa adicional.

A pesar de que existen númerosos lenguajes de scripting y la mayoría considerablemente más versátiles y potentes que NASL, ninguno de ellos es veraderamente seguro si se tiene en cuenta que cualquiera puede ser usado para escribir un troyano con capacidad para abrir una conexión a un tercero, al que se permitiría saber que dicha máquina está funcionando como un servidor Nessus e incluso los nombres de los sistemas que están siendo objeto de análisis.

Por otro lado, NASL se caracteriza por ser un lenguaje con un consumo de recursos muy limitado ya que está optimizado para un tipo de interacción en red muy específica. Sería posible, por ejemplo, lanzar hasta 20 instancias del servidor *nessusd* en la misma máquina y de forma concurrente sin que ésta tenga la necesidad de alcanzar los 256Mb de RAM instalados [4].

6.2.3. Auditorias de Configuración

Además del análisis de vulnerabilidades, Nessus puede ser utilizado para efectuar auditorías de configuración sobre todos los *hosts* de una red corporativa de acuerdo con una determinada directiva de seguridad. Si la organización a la que pertenece la red posee una directiva específica para la configuración de los servidores o si se desea evaluar el nivel de conformidad de los sistemas respecto a guías o recomendaciones de seguridad públicas o gubernamentales como NSA, CERT o CIS, Nessus puede realizar un análisis tanto de sistemas UNIX como Windows de forma automática. Para los sistemas Windows pueden ser verificados políticas de usuario, permisos de fichero, permisos de acceso al registro, permisos de ejecución de servicios o determinados eventos. Para sistemas UNIX, políticas de usuario, permisos de fichero, procesos en ejecución y comprobaciones de contenido de ficheros.

Cuando la auditoría de la red se realiza en base al soporte profesional del Centro de Seguridad Tenable, la empresa puede realizar un análisis integral de vulnerabilidades y configuración. De este modo, las organizaciones pueden dar recomendaciones a las partes responsables y verificar la instalación de parches de seguridad y la conformidad respecto a una configuración predefinida. Algunas políticas de seguridad cuyo cumplimiento puede ser verificado son:

- Guías de buenas prácticas CIS (*Center for Internet Security*)
- DISA STIGs (*Defense Information Systems Agency Security Technical Implementation Guides*)
- Contenidos NIST SCAP (*National Institute of Standards and Technology*)
- Guias de buenas prácticas NSA (*American National Security Agency*)
- Requerimientos de configuración PCI (*Payment Card Industry*)
- Recomendaciones de configuración de proveedores

Los plugins que permiten realizar este tipo de auditoría se encuentran disponibles a través del *feed profesional*.

Por último, Nessus también puede ser utilizado para realizar auditorías de contenido en sistemas Windows, ya que puede acceder a los diferentes *hosts* con el fin de localizar información sensible para la empresa y susceptible de ser sustraida por terceros [6].

6.2.4. Actualizaciones

El sistema de actualización de Nessus está basado en dos modelos de gestión y adquisición de plugins por parte de los usuarios: el *HomeFeed* y el *ProfessionalFeed*. Aunque Nessus ha sido una herramienta *open source* hasta 2005, ha sido en 2008 cuando se ha retirado el conocido como *"registered feed"*, un feed de libre acceso que podía ser utilizado tanto para uso personal como corporativo y es ahora su sustituto, el *HomeFeed*, el que se encuentra disponible, aunque solamente con licencia para uso en una red particular. El *ProfessionalFeed*, sólo accesible mediante suscripción de pago, permite utilizar los plugins adquiridos para análisis de redes corporativas o de uso comercial. A través de este tipo de suscripción el usuario puede disponer no sólo de las actualizaciones de plugins sino también de soporte técnico por parte de la empresa Tenable y su *Security Center*.

Al margen de la asistencia técnica, la posibilidad de llevar a cabo auditorías de configuración como parte de análisis programados y otras opciones, como acceso a herramientas de virtualización para comprobación del funcionamiento de Nessus en entornos virtuales (Nessus 3 VMware Virtual Appliance, el cual funciona con VMware ESX, Server, Workstation y Fusion), no existe diferencia entre los plugins disponibles por medio de uno u otro feed. Esto es, la información sobre las nuevas vulnerabilidades y la disponibilidad de sus correspondientes exploits utilizables como plugins de Nessus [6].

Nessus se actualiza de forma constante y en el momento de redacción de este documento cuenta con 24243 plugins disponibles. Cada semana se añaden varias docenas de plugins y una vez que el escaner ha sido registrado por su usuario en *Nessus.org*, éste se actualiza de forma automática cada 24 horas cuando la opción `auto_update` tiene el valor `yes` en el archivo de configuración /opt/nessus/etc/nessus/nessusd.conf (esta es la opción por defecto). Para determinar si el escaner ha sido correctamente registrado, el siguiente comando:

```
$>/opt/nessus/bin/nessus-fetch -check
```

debería devolver el siguiente mensaje:

```
nessus-fetch is properly configured to receive a Home feed
```

para el caso de un registro con el *HomeFeed*. Mediante el comando *nessus-update-plugins* es posible forzar una actualización no automática en cualquier momento. Generalmente, *Tenable Network Security*, la empresa propietaria de Nessus, escribe y libera plugins nuevos a las 24 horas de ser hechas públicas las vulnerabilidades. Los plugins, codificados en NASL, contienen información de la vulnerabilidad, un conjunto básico de soluciones a la misma y el código del exploit que permite detectar su presencia en un sistema. En algunos casos, un plugin puede depender de otro plugin.

De todos los plugins disponibles, 9449 poseen identificadores CVE (Commun Vulnerabilities Exposure) y 6773 poseen identificadores Bugtraq únicos. Los identificadores CVE son un estándar para los nombres de las vulnerabilidades, permiten el intercambio de datos sobre las mismas entre diferentes productos y proporcionan un índice de base para evaluar la cobertura de distintas herramientas. Tenable sincroniza de forma diaria las entradas CVE de Nessus con la información de la *National Vulnerability Database*. Si existe una entrada CVE nueva o corregida, el contenido disponible para Nessus se modifica de forma acorde. Los identificadores CVE están gestionados por la corporación *Mitre*, una organización sin ánimo de lucro y subvencionada por el departamento estadounidense de seguridad nacional [1]. En el caso de los identificadores Bugtraq, éstos pertenecen a la lista de correo Bugtraq, alojada en *SecurityFocus.com* y perteneciente a la empresa *Symantec*. En esta lista de correo, una de las más activas y con más volumen del mundo, se generan discusiones sobre vulnerabilidades, se anuncian fallos de seguridad, métodos de explotación de dichas vulnerabilidades y como solucionarlas. La gran mayoría de las mismas son discutidas en este lista y se les asigna un identificador y descripción únicos, de ahi que éste también represente un buen indicador para evaluar el alcance de distintos sistemas de detección de vulnerabilidades [8]. Nessus tiene en cuenta además otros tipos de estándares como CVSS o XCCDF [6][2].

Respecto a los periodos de desfase entre la aparición de una vulnerabilidad y la disponibilidad de su plugin correspondiente, en la mayoría de los casos se puede disponer del mismo en un tiempo medio de 24 horas tras la aparición del exploit. Este tiempo es el mismo tanto para suscriptores corporativos como no comerciales. En cualquier caso, Nessus está considerado una de las aplicaciones de análisis con tiempos de actualizacion más rápidos y su efectividad

no dependerá tanto del tiempo de actualización como de la cantidad de vulnerabilidades que permita identificar respecto a otra aplicación.

6.3. Pruebas de concepto

Las reglas utilizadas para llevar a cabo las siguientes pruebas y que constituyen el motor de detección de Snort, son las publicadas el 16 de Mayo de 2009 para usuarios con suscripción y el 16 de Junio de 2009 para usuarios solamente registrados. Se entiende, por tanto, que la respuesta del IDS es la que tendría un sistema actualizado por usuarios de pago a 16 de Mayo de 2009 y por usuarios registrados, a 16 de Junio de 2009.

El escáner Nessus se encuentra actualizado a 15 de julio por lo que se supone capaz de probar muchas de las vulnerabilidades presentes en los servicios instalados, la mayoría de ellos anteriores a 2008 y para los que existen actualizaciones de seguridad o versiones actualizadas no instaladas.

Para las diferentes pruebas de concepto, se han probado los plugins existentes para cada uno de estos servicios y una vez identificadas las vulnerabilidades que Nessus es capaz de detectar en cada uno de ellos, se han utilizado dichos plugins uno a uno para verificar la respuesta de Snort a cada una de esas pruebas.

El objetivo es comprobar la existencia tanto de falsos negativos, ya sean porque no se identifica el ataque correctamente o porque ni siquiera se está generando una alerta, como de falsos positivos, habitualmente aquellos casos en los que una prueba genera tráfico que si bien no permite a Snort detectar un ataque a una vulnerabilidad concreta, lo hace generar alertas de otro tipo o para otros ataques.

La selección de servidores escogida e instalada responde al interés de establecer un entorno de pruebas habitual en sistemas reales donde se encuentran accesibles estos servicios de uso extendido. En la mayoría de los casos no es necesario dar a Nessus acceso al sistema, con lo que podrían ser explotados sin más conocimiento que la existencia de que el puerto está abierto. Por otro lado, si se intenta un ataque *knowledge-based*, introduciendo en la configuración de Nessus un usuario y una contraseña reales del sistema, el número de vulnerabilidades que se detecta es mayor, y aunque para una auditoría seria recomendable incluir este paso como parte de la misma, para un atacante, implicaría conocer el login y el pass de algún usuario.

6.3.1. Vulnerabilidades en servidor Apache

El servidor Apache es un servidor HTTP de código abierto y el más popular en Internet desde 1996. Existen versiones para sistemas UNIX y Microsoft Windows. Para esta prueba de concepto se ha instalado el paquete *apache-1.3.34-2ubuntu0.1* en el sistema objetivo y el servidor se encuentra activo en el puerto 80.

Tras una prueba inicial con todos los plugins asociados al servidor Apache en sistemas Ubuntu Linux, el escáner Nessus detecta la vulnerabilidad crítica con identificador CVE-2006-3747. La figura 6.1 muestra parte del resultado asociado a dicha vulnerabilidad una vez realizada una prueba en la que sólo se encuentra activo el plugin asociado. En este caso Nessus detecta la versión del servidor y a partir de su base de datos identifica la vulnerabilidad del software.

La figura 6.2 muestra las alertas generadas por Snort en la interfaz BASE.

Comprobamos que se han generado dos alertas significativas:

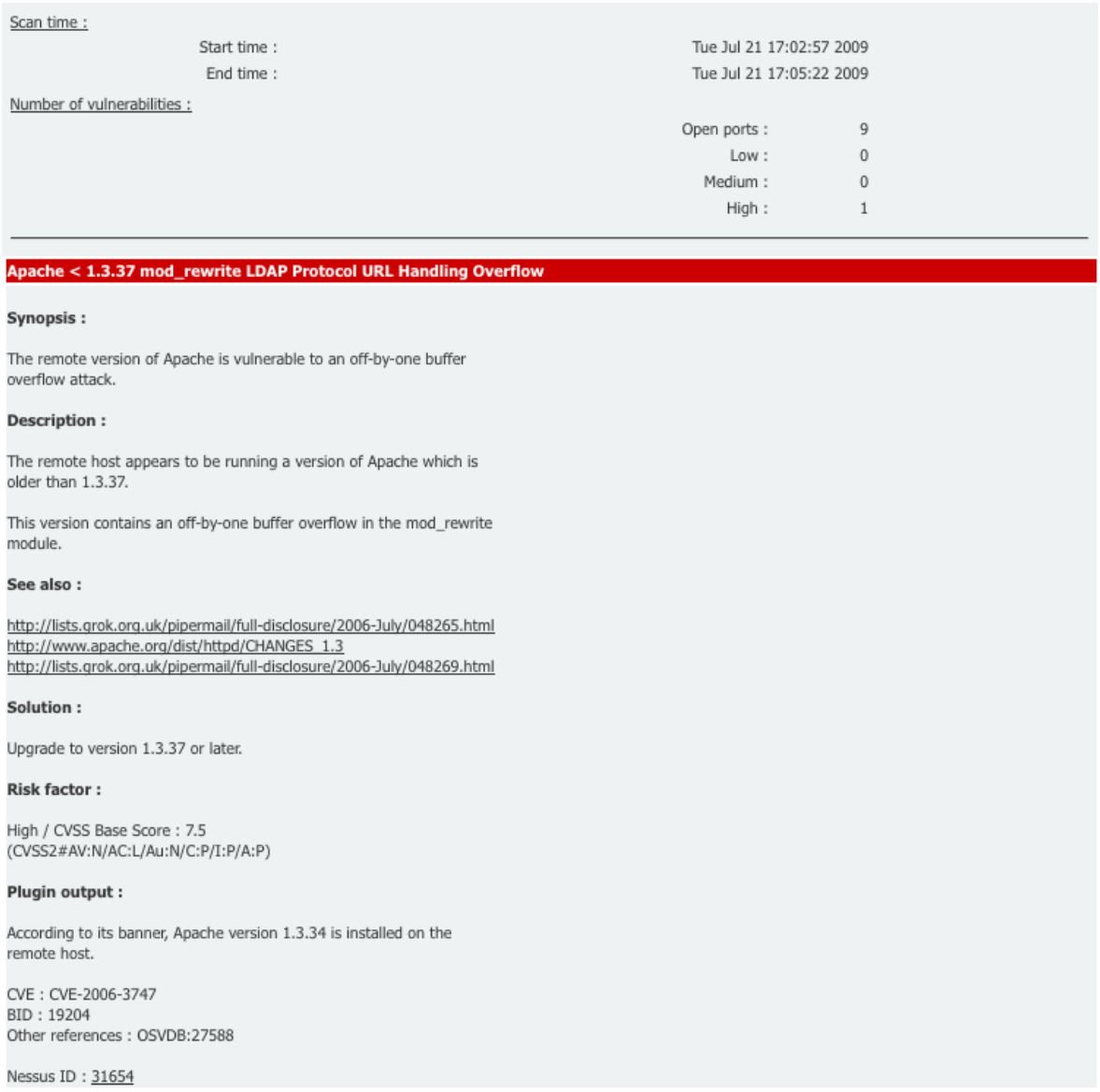
```
2585 || WEB-MISC nessus 2.x 404 probe || nessus,10386
```

La regla 2585 genera una alerta cuando el plugin de Nessus 10386 intenta comprobar si el servidor Apache devuelve o no códigos de error 404 cuando se solicita un archivo que no se encuentra disponible.

```
1563 || WEB-MISC login.htm attempt || bugtraq,665 || cve,1999-1533
```

La regla 1563 genera una alerta cuando un usuario intenta provocar una denegación de servicio enviando un password largo al fichero de acceso login.htm.

En este caso, comprobamos que Nessus ha realizado una petición de acceso al servidor a partir de la cual identificar la versión del mismo. El IDS Snort no genera falsos negativos ya que ningún exploit es enviado al servidor pero si identifica el acceso con autenticación como una posible amenaza, lo cual puede ser entendido como un falso positivo.



Scan time :

Start time :	Tue Jul 21 17:02:57 2009
End time :	Tue Jul 21 17:05:22 2009

Number of vulnerabilities :

Open ports :	9
Low :	0
Medium :	0
High :	1

Apache < 1.3.37 mod_rewrite LDAP Protocol URL Handling Overflow

Synopsis :

The remote version of Apache is vulnerable to an off-by-one buffer overflow attack.

Description :

The remote host appears to be running a version of Apache which is older than 1.3.37.

This version contains an off-by-one buffer overflow in the mod_rewrite module.

See also :

http://lists.grok.org.uk/pipermail/full-disclosure/2006-July/048265.html
http://www.apache.org/dist/httpd/CHANGES_1.3
http://lists.grok.org.uk/pipermail/full-disclosure/2006-July/048269.html

Solution :

Upgrade to version 1.3.37 or later.

Risk factor :

High / CVSS Base Score : 7.5
(CVSS2#AV:N/AC:L/Au:N/C:P/I:P/A:P)

Plugin output :

According to its banner, Apache version 1.3.34 is installed on the remote host.

CVE : CVE-2006-3747
BID : 19204
Other references : OSVDB:27588

Nessus ID : 31654

Figura 6.1: Resultados del escáner Nessus en la PoC #1

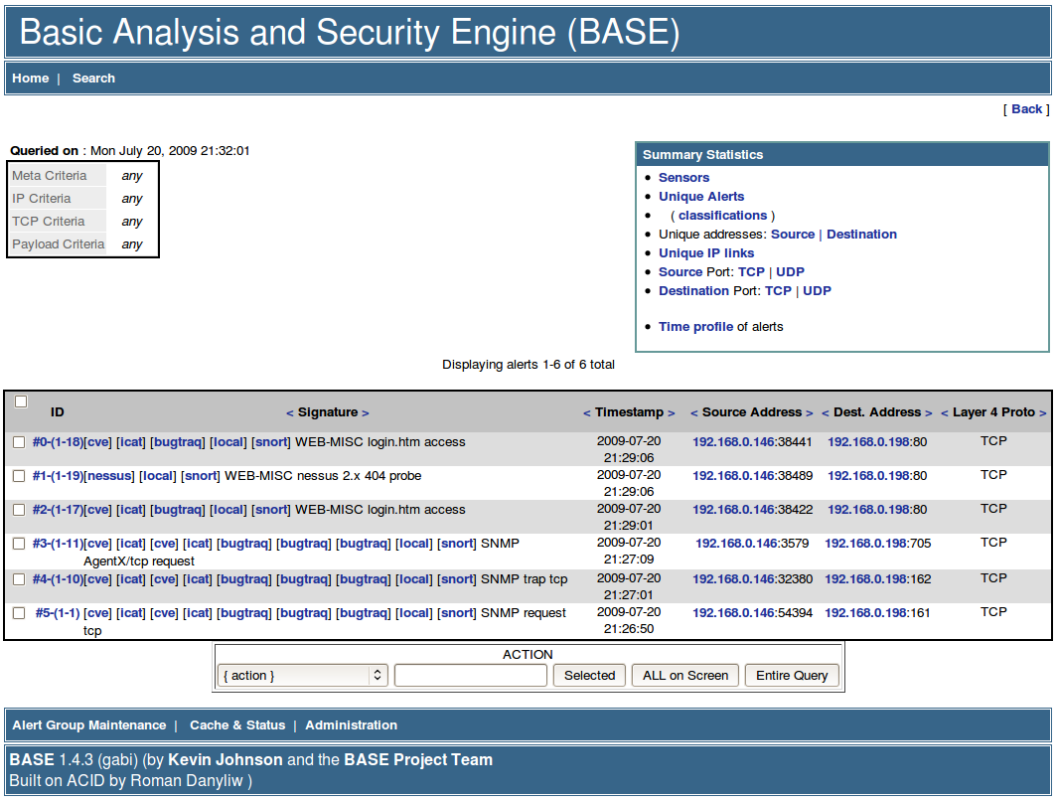


Figura 6.2: Alertas generadas por Snort en la interfaz BASE para la PoC #1

6.3.2. Vulnerabilidades en servidor FTP

El servidor FTP es un servicio que implementa el protocolo de transferencia de ficheros (FTP). El servicio FTP es ofrecido por la capa de aplicación del modelo de capas de red TCP/IP al usuario y puede ser accedido por un cliente para descargar o subir archivos independientemente del sistema operativo utilizado en ambos equipos. Para esta prueba de concepto se ha instalado el paquete *vsftpd-2.0.4-0ubuntu4, The Very Secure FTP Daemon*, en el sistema objetivo y el servidor se encuentra activo en el puerto 21.

Tras una prueba inicial con todos los plugins asociados al servidor FTP en sistemas Ubuntu Linux, el escáner Nessus detecta una vulnerabilidad de riesgo medio con identificador CVE-1999-0497. La figura 6.3 muestra parte del resultado asociado a dicha vulnerabilidad una vez realizada una prueba en la que sólo se encuentra activo el plugin asociado. En este caso Nessus detecta que es posible acceder al servidor con un login de usuario anónimo, lo que puede permitir el acceso a un usuario no autorizado a información sensible.


Scan time :

Start time : Mon Jul 20 23:57:43 2009
End time : Tue Jul 21 00:00:34 2009

Number of vulnerabilities :

Open ports : 10
Low : 0
Medium : 1
High : 0

Anonymous FTP Enabled

Synopsis :

Anonymous logins are allowed on the remote FTP server.

Description :

This FTP service allows anonymous logins. If you do not want to share data with anyone you do not know, then you should deactivate the anonymous account, since it can only cause troubles.

Risk factor :

Medium / CVSS Base Score : 5.0
(CVSS2#AV:N/AC:L/Au:N/C:P/I:N/A:N)

CVE : CVE-1999-0497
Other references : OSVDB:69

Nessus ID : 10079

Figura 6.3: Resultados del escáner Nessus en la PoC #2

La figura 6.4 muestra las alertas de tipo TCP generadas por Snort en la interfaz BASE.

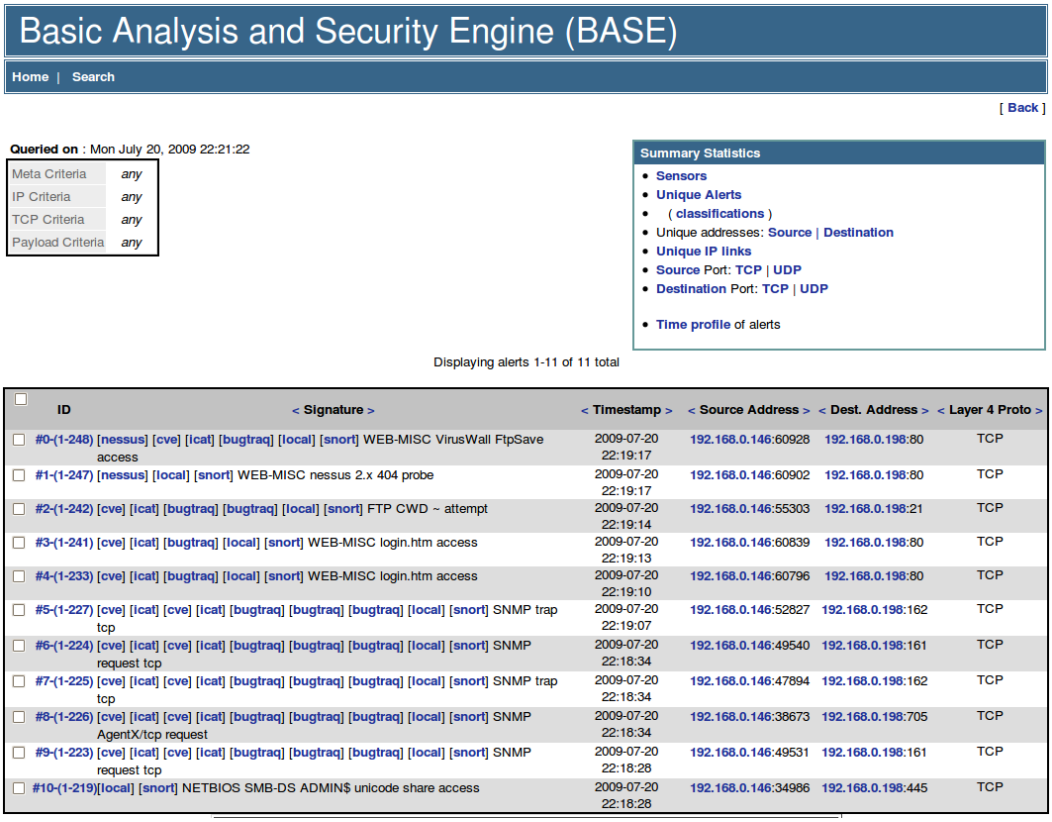


Figura 6.4: Alertas de tipo TCP generadas por Snort en la interfaz BASE para la PoC #2

Se han generado tres alertas significativas de tipo TCP relacionadas con el acceso FTP:

```
1230 || WEB-MISC VirusWall FtpSave access || bugtraq,2808
|| cve,2001-0432 || nessus,10733
```

La regla 1230 genera una alerta cuando un atacante intenta acceder al servicio FTP del scáner VirusWall, el cual puede ser modificado de forma remota por un atacante sin autorización usando peticiones construidas maliciosamente. Ya que este sistema no se encuentra instalado, se esta produciendo, por falta de contexto, un falso positivo.

```
1672 || FTP CWD ~ attempt || bugtraq,2601 || bugtraq,9215 || cve,2001-0421
```

La regla 1672 genera una alerta cuando un atacante intenta acceder al sistema mediante un usuario válido, en este caso, el usuario anónimo y un password incorrecto seguido de *CWD command*, lo que podría permitir acceder a información sensible.

```
2474 || NETBIOS SMB-DS ADMIN$ share access
```

La regla 2474 genera una alerta cuando se produce un acceso a un elemento compartido a través de un servidor Samba, de modo que en este caso se ha producido un falso positivo.

Al igual que en la prueba del servidor Apache, se han registrado accesos al fichero login.htm y un intento de comprobar si se genera un código de error 404 cuando no se encuentra un fichero solicitado. Tanto estas, como las alertas generadas a partir de paquetes SNMP deberían ser interpretadas como falsos positivos.

La figura 6.5 muestra las alertas de tipo UDP generadas por Snort en la interfaz BASE.

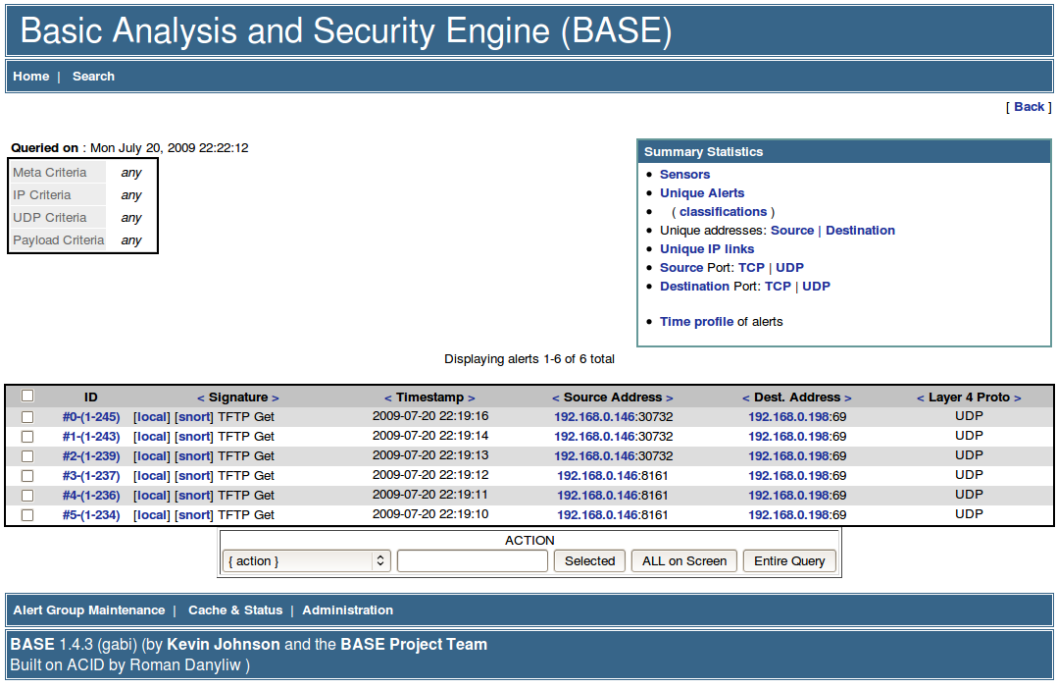


Figura 6.5: Alertas de tipo UDP generadas por Snort en la interfaz BASE para la PoC #2

Comprobamos que se ha generado una alerta de tipo UDP relacionada con el acceso al servidor FTP en repetidas ocasiones:

```
1444 || TFTP Get
```

Todas las alertas generadas por la regla 1444 responden simplemente a peticiones de tipo GET al servicio FTP. En el caso de existir un servidor real al que los usuarios puedan acceder, un acceso de tipo GET se corresponde con una actividad normal y este tipo de alertas podrían ser desactivadas considerándose falsos positivos.

6.3.3. Vulnerabilidades en servidor IMAP

El servidor IMAP es un servidor de correo que implementa el protocolo *Internet Message Access Protocol* y permite a los usuarios acceder y modificar los mensajes de correo almacenados en la máquina donde se encuentra disponible dicho servidor. Para esta prueba de concepto se ha instalado el paquete *courier-imap-3.0.8-13ubuntu5* en el sistema objetivo y el servidor se encuentra activo en el puerto TCP 143.

Tras una prueba inicial con todos los plugins asociados al servidor IMAP en sistemas Ubuntu Linux, el escáner Nessus detecta tan sólo una vulnerabilidad de riesgo bajo al poder obtener el banner del servidor y con ello información acerca del servicio, que puede ser utilizada para generar un ataque más específico. La figura 6.6 muestra parte del resultado asociado a dicha vulnerabilidad una vez realizada una prueba en la que sólo se encuentra activo el plugin asociado.


Scan time :

Start time : Tue Jul 21 18:41:21 2009

End time : Tue Jul 21 18:42:12 2009

Number of vulnerabilities :

Open ports : 10

Low : 1

Medium : 0

High : 0

IMAP Service Banner Retrieval

Synopsis :

An IMAP server is running on the remote host.

Description :

An IMAP (Internet Message Access Protocol) server is installed and running on the remote host.

Risk factor :

None

Plugin output :

The remote imap server banner is :

* OK [CAPABILITY IMAP4rev1 UIDPLUS CHILDREN NAMESPACE THREAD=ORDEREDSUBJECT THREAD=REFERENCES SORT QUOTA IDLE ACL ACL2=UNION] Courier-IMAP ready. Copyright 1998-2004 Double Precision, Inc. See COPYING for distribution information.

Nessus ID : 11414

Figura 6.6: Resultados del escáner Nessus en la PoC #3

La figura 6.7 muestra las alertas generadas por Snort en la interfaz BASE.

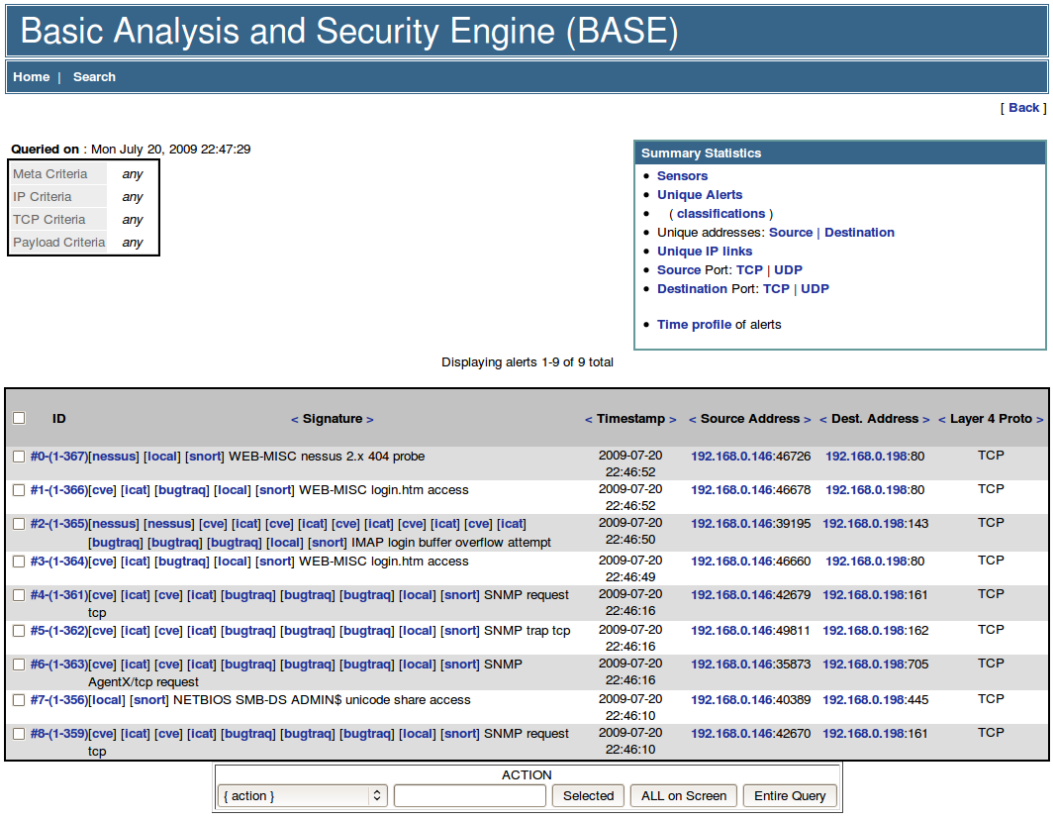


Figura 6.7: Alertas generadas por Snort en la interfaz BASE para la PoC #3

Comprobamos que se ha generado sólo una alerta relacionada con el acceso al servidor IMAP:

```
1842 || IMAP login buffer overflow attempt || bugtraq,502
|| cve,1999-0005 || cve,1999-1557 || nessus,10123 || nessus,10125
```

La regla 1842 genera este tipo de alerta cuando se detecta que un atacante intenta aprovechar un buffer overflow remoto en el servidor IMAP mediante el envío de un login lo suficientemente largo como para provocar el fallo. Aunque según su descripción, sólo los plugins 10123 y 10125 de Nessus, los cuales se encontraban desactivados, realizan esta comprobación, es posible que el plugin 11414 que detecta la presencia del servidor, haya podido generar dicha alerta al intentar acceder al servicio, de modo que la alerta puede interpretarse como un falso positivo.

6.3.4. Vulnerabilidades en servidor MySQL

El servidor MySQL es una implementación de código abierto de un sistema de gestión de base de datos relacional, multihilo y multiusuario propiedad de la empresa Sun Microsystems. MySQL es muy popular como base de datos asociada a servicios WEB por su sencilla integración con el lenguaje PHP, muy utilizado en este tipo de aplicaciones. Para esta prueba de concepto se ha instalado el paquete *mysql-server-5.0.21-3ubuntu1* en el sistema objetivo y el servidor se encuentra activo en el puerto TCP 3306.

Tras una prueba inicial con todos los plugins asociados al servidor MySQL en sistemas Ubuntu Linux, el escáner Nessus detecta varias vulnerabilidades definidas con riesgo medio y con identificadores CVE-2007-3780, CVE-2007-3781, CVE-2007-3782.

La figura 6.8 muestra parte del resultado una vez realizada una prueba en la que sólo se encuentra activo el plugin asociado a dichas vulnerabilidades.

La figura 6.9 muestra las alertas generadas por Snort en la interfaz BASE.

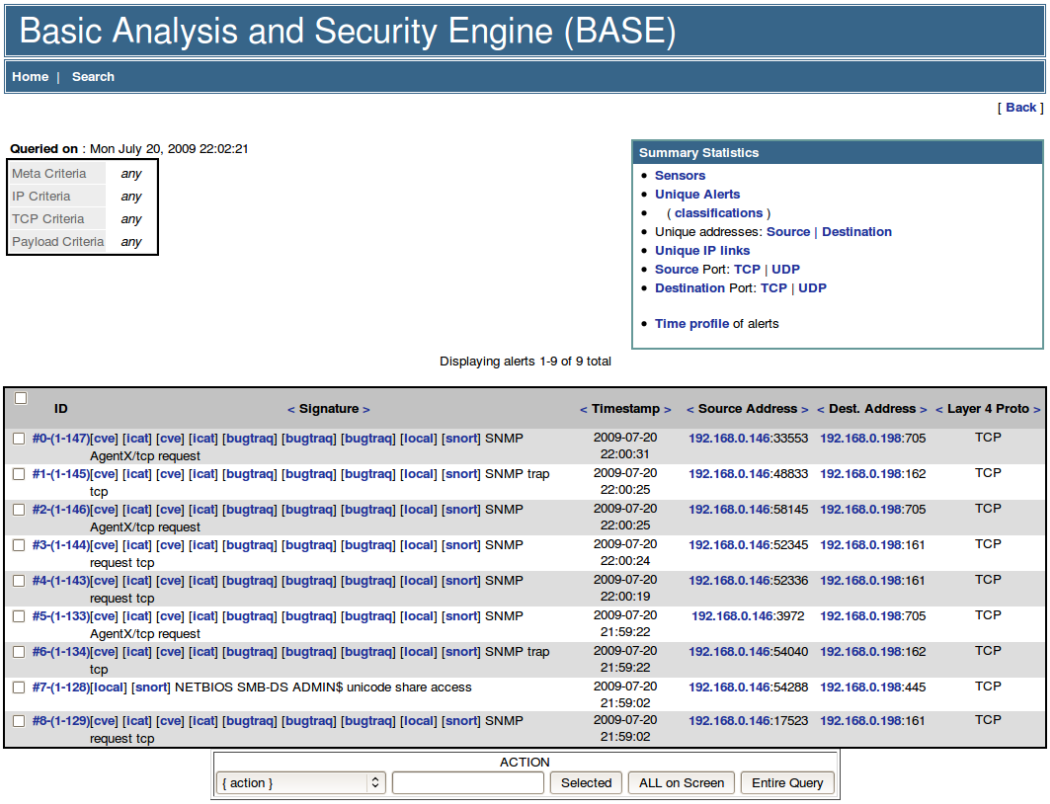
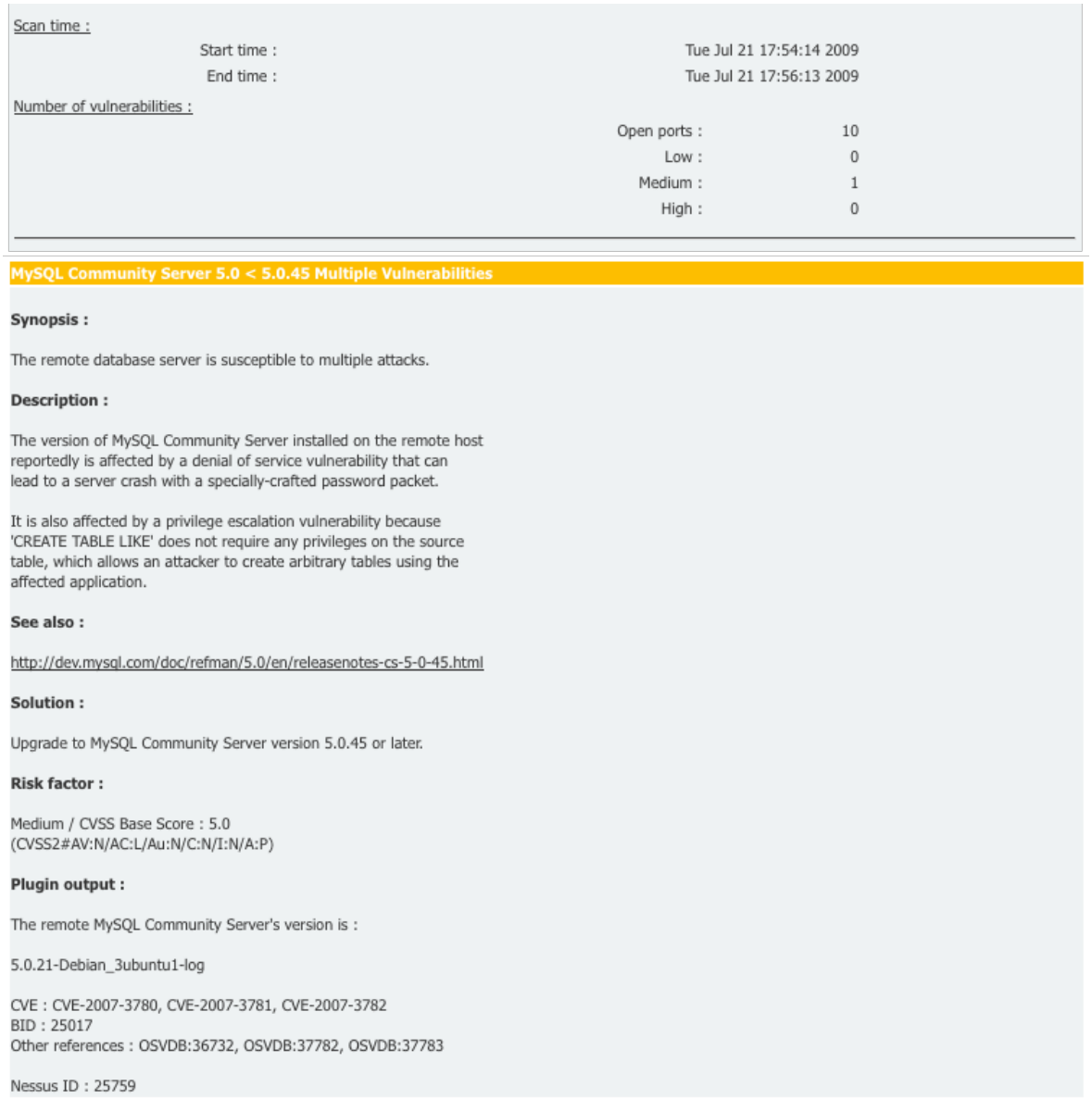


Figura 6.9: Alertas generadas por Snort en la interfaz BASE para la PoC #4



Scan time :

Start time :	Tue Jul 21 17:54:14 2009
End time :	Tue Jul 21 17:56:13 2009

Number of vulnerabilities :

Open ports :	10
Low :	0
Medium :	1
High :	0

MySQL Community Server 5.0 < 5.0.45 Multiple Vulnerabilities

Synopsis :

The remote database server is susceptible to multiple attacks.

Description :

The version of MySQL Community Server installed on the remote host reportedly is affected by a denial of service vulnerability that can lead to a server crash with a specially-crafted password packet.

It is also affected by a privilege escalation vulnerability because 'CREATE TABLE LIKE' does not require any privileges on the source table, which allows an attacker to create arbitrary tables using the affected application.

See also :

http://dev.mysql.com/doc/refman/5.0/en/releasenotes-cs-5-0-45.html

Solution :

Upgrade to MySQL Community Server version 5.0.45 or later.

Risk factor :

Medium / CVSS Base Score : 5.0
(CVSS2#AV:N/AC:L/Au:N/C:N/I:N/A:P)

Plugin output :

The remote MySQL Community Server's version is :

5.0.21-Debian_3ubuntu1-log

CVE : CVE-2007-3780, CVE-2007-3781, CVE-2007-3782
BID : 25017
Other references : OSVDB:36732, OSVDB:37782, OSVDB:37783

Nessus ID : 25759

Figura 6.8: Resultados del escáner Nessus en la PoC #4

Comprobamos que en este caso Snort no genera ninguna alerta asociada al servidor MySQL, sin embargo, Nessus es capaz de acceder y detectar que la versión instalada es susceptible de sufrir un ataque de denegación de servicio y una escalada de privilegios, ya que 'CREATE TABLE LIKE' no requiere permisos especiales y puede permitir a un atacante crear tablas arbitrarias.

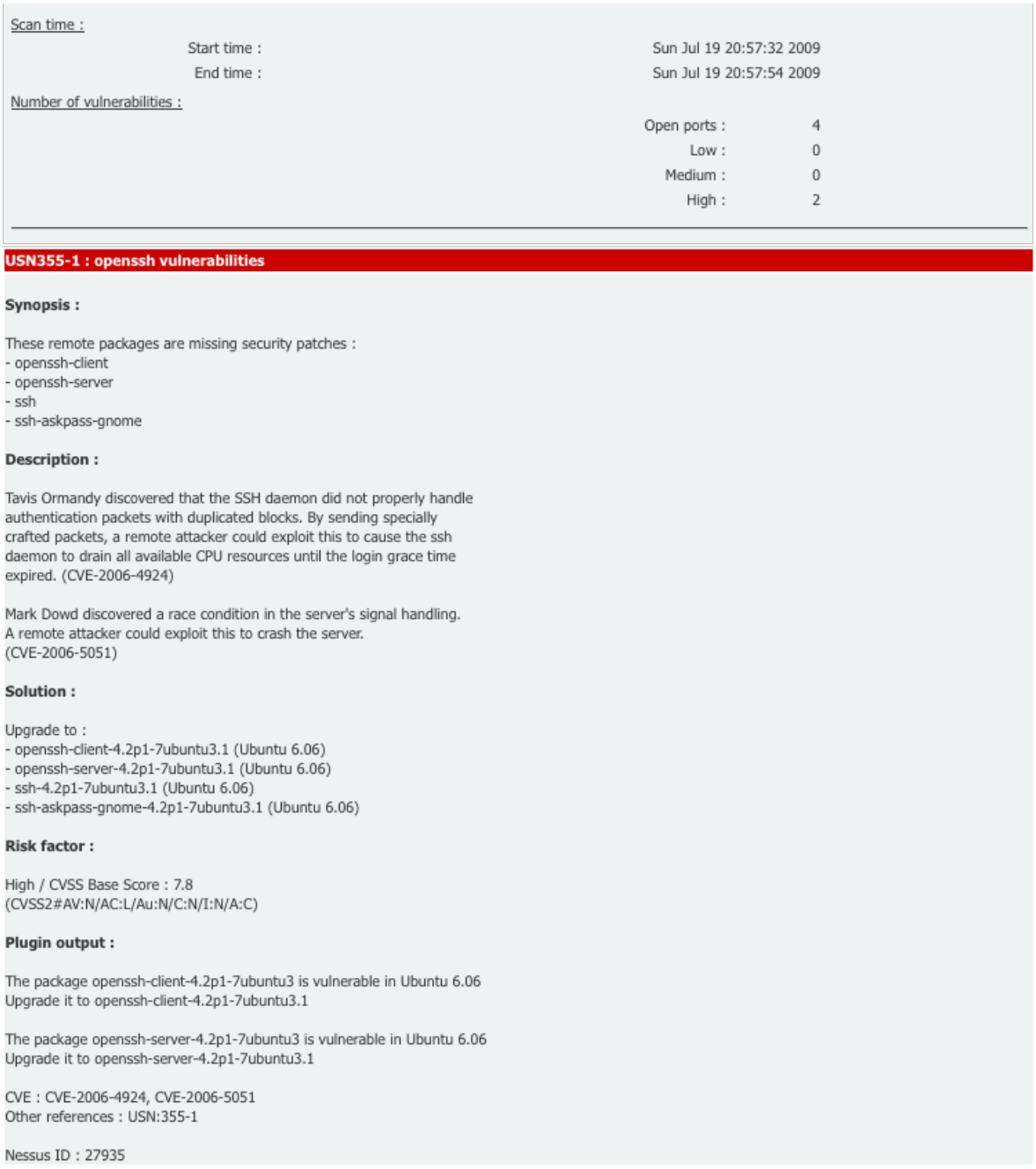
En esta prueba de concepto no se está enviando ningún exploit o paquete mal formado, no obstante, la falta de información sobre el contexto de aplicación puede llevar a accesos no autorizados al sistema y a posteriores escaladas de privilegios que el IDS no es capaz de detectar.

6.3.5. Vulnerabilidades en OpenSSH

OpenSSH (Open Secure Shell) es un conjunto de aplicaciones que permiten realizar comunicaciones cifradas a través de una red, usando el protocolo SSH. Fue creado como una alternativa libre y abierta al programa Secure Shell, que es software propietario [19]. Para esta prueba de concepto se ha instalado el paquete *openssh-server-4.2p1-7ubuntu3* en el sistema objetivo y el servidor se encuentra activo en el puerto TCP 22.

Tras una prueba inicial con todos los plugins asociados al servidor MySQL en sistemas Ubuntu Linux, el escáner Nessus detecta varias vulnerabilidades asociadas a dos plugins de Nessus, definidas con riesgo alto y con identificadores CVE-2006-4924, CVE-2006-5051, CVE-2008-0166.

La figura 6.10 muestra la primera parte del resultado una vez realizada una prueba en la que sólo se encuentran activos los plugin asociados a dichas vulnerabilidades. El plugin 27935 definido como *USN355-1: OpenSSH vulnerabilities* detecta que varios paquetes del conjunto de aplicaciones OpenSSH requieren actualizaciones de seguridad y que el servicio SSH no es capaz de manejar paquetes de autenticación con bloques duplicados. Un atacante remoto puede provocar una denegación de servicio por bloqueo de la CPU mediante envío de un paquete especialmente formado. También existe la posibilidad de terminar el servidor aprovechando un fallo en el manejo de señales provocado por una condición de carrera.



Scan time :

Start time :	Sun Jul 19 20:57:32 2009
End time :	Sun Jul 19 20:57:54 2009

Number of vulnerabilities :

Open ports :	4
Low :	0
Medium :	0
High :	2

USN355-1 : openssh vulnerabilities

Synopsis :

These remote packages are missing security patches :
- openssh-client
- openssh-server
- ssh
- ssh-askpass-gnome

Description :

Tavis Ormandy discovered that the SSH daemon did not properly handle authentication packets with duplicated blocks. By sending specially crafted packets, a remote attacker could exploit this to cause the ssh daemon to drain all available CPU resources until the login grace time expired. (CVE-2006-4924)

Mark Dowd discovered a race condition in the server's signal handling. A remote attacker could exploit this to crash the server. (CVE-2006-5051)

Solution :

Upgrade to :
- openssh-client-4.2p1-7ubuntu3.1 (Ubuntu 6.06)
- openssh-server-4.2p1-7ubuntu3.1 (Ubuntu 6.06)
- ssh-4.2p1-7ubuntu3.1 (Ubuntu 6.06)
- ssh-askpass-gnome-4.2p1-7ubuntu3.1 (Ubuntu 6.06)

Risk factor :

High / CVSS Base Score : 7.8
(CVSS2#AV:N/AC:L/Au:N/C:N/I:N/A:C)

Plugin output :

The package openssh-client-4.2p1-7ubuntu3 is vulnerable in Ubuntu 6.06
Upgrade it to openssh-client-4.2p1-7ubuntu3.1

The package openssh-server-4.2p1-7ubuntu3 is vulnerable in Ubuntu 6.06
Upgrade it to openssh-server-4.2p1-7ubuntu3.1

CVE : CVE-2006-4924, CVE-2006-5051
Other references : USN:355-1

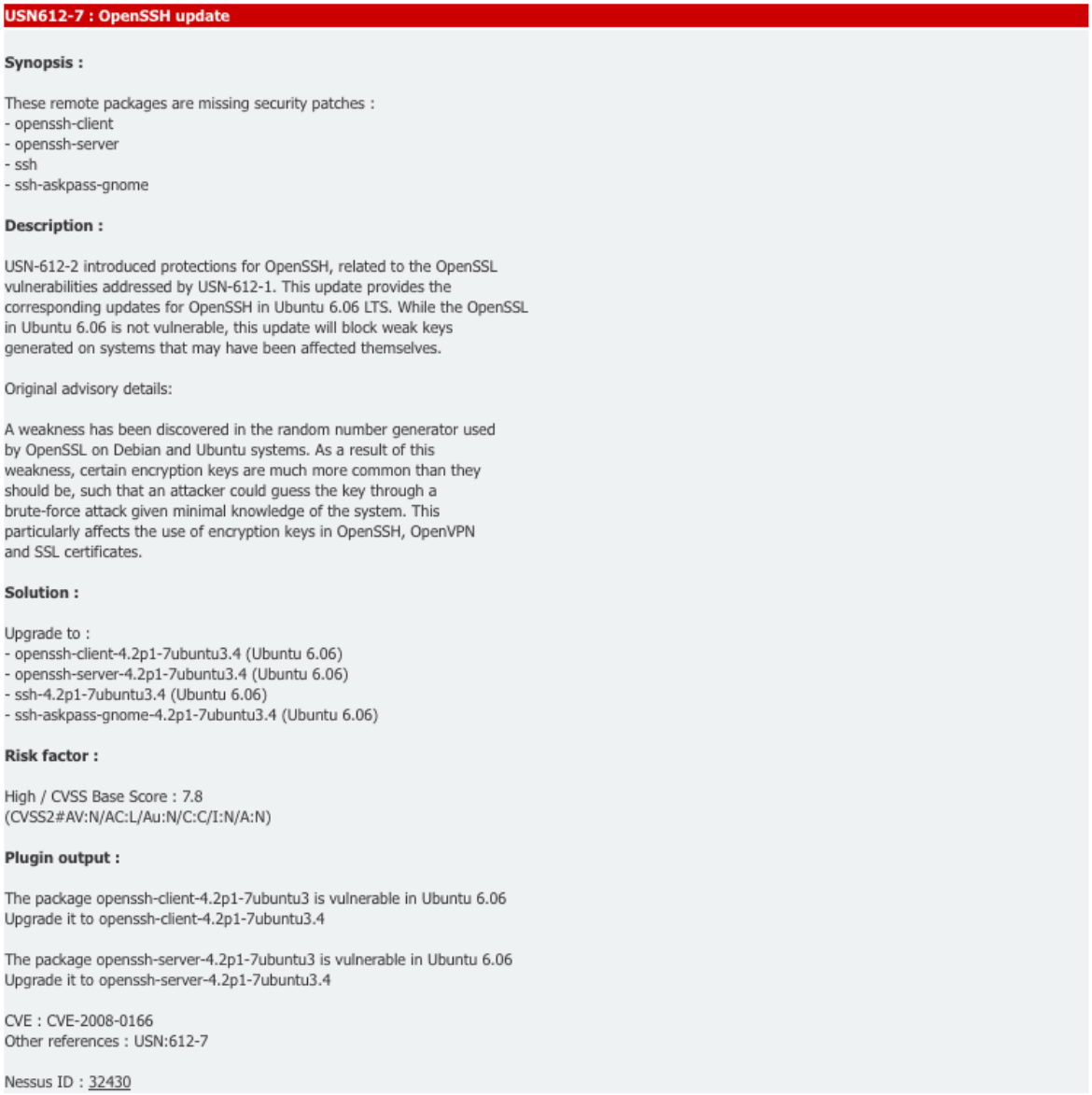
Nessus ID : 27935

Figura 6.10: Resultados #1 del escáner Nessus en la PoC #5

La figura 6.11 muestra el resultado obtenido por el plugin 32430 definido como *USN612-7 OpenSSH update*, el cual detecta también que varios paquetes de la aplicación requieren actualizaciones de seguridad que no se encuentran instaladas. La actualización requerida corrige un fallo en el generador de números aleatorios utilizado por OpenSSL en sistemas Debian y

Ubuntu y que tiene como resultado que algunas claves sean mas probables que otras. Esto puede ser aprovechado por un atacante para realizar un ataque por fuerza bruta con un conocimiento mínimo del sistema. Este fallo afecta a la generación de claves en OpenSSH y a los certificados SSL.

En este caso particular, no existe una vulnerabilidad de red como tal y el fallo es descubierto mediante la comparación de versiones por lo que no es de esperar una detección por parte del IDS a falta de un contexto de aplicación.



USN612-7 : OpenSSH update

Synopsis :

These remote packages are missing security patches :
- openssh-client
- openssh-server
- ssh
- ssh-askpass-gnome

Description :

USN-612-2 introduced protections for OpenSSH, related to the OpenSSL vulnerabilities addressed by USN-612-1. This update provides the corresponding updates for OpenSSH in Ubuntu 6.06 LTS. While the OpenSSL in Ubuntu 6.06 is not vulnerable, this update will block weak keys generated on systems that may have been affected themselves.

Original advisory details:

A weakness has been discovered in the random number generator used by OpenSSL on Debian and Ubuntu systems. As a result of this weakness, certain encryption keys are much more common than they should be, such that an attacker could guess the key through a brute-force attack given minimal knowledge of the system. This particularly affects the use of encryption keys in OpenSSH, OpenVPN and SSL certificates.

Solution :

Upgrade to :
- openssh-client-4.2p1-7ubuntu3.4 (Ubuntu 6.06)
- openssh-server-4.2p1-7ubuntu3.4 (Ubuntu 6.06)
- ssh-4.2p1-7ubuntu3.4 (Ubuntu 6.06)
- ssh-askpass-gnome-4.2p1-7ubuntu3.4 (Ubuntu 6.06)

Risk factor :

High / CVSS Base Score : 7.8
(CVSS2#AV:N/AC:L/Au:N/C:C/I:N/A:N)

Plugin output :

The package openssh-client-4.2p1-7ubuntu3 is vulnerable in Ubuntu 6.06
Upgrade it to openssh-client-4.2p1-7ubuntu3.4

The package openssh-server-4.2p1-7ubuntu3 is vulnerable in Ubuntu 6.06
Upgrade it to openssh-server-4.2p1-7ubuntu3.4

CVE : CVE-2008-0166
Other references : USN:612-7

Nessus ID : 32430

Figura 6.11: Resultados #2 del escáner Nessus en la PoC #5

La figura 6.12 muestra las alertas generadas por Snort en la interfaz BASE. Se comprueba que no se ha generado ningún tipo de alerta en referencia al servidor OpenSSH. Aunque en el caso del segundo fallo no se trata propiamente de una vulnerabilidad de red, las primeras vulnerabilidades detectadas si que pueden ser explotadas mediante exploits enviados de forma remota. La detección realizada por Nessus se basa, en este caso, en la comparación de versiones y no en un intento de explotación real. Por tanto, podemos considerar que no se han producido falsos negativos en la detección pero quedaría pendiente comprobar si un intento de aprovechar dichos fallos con paquetes mal formados sería detectado por Snort.

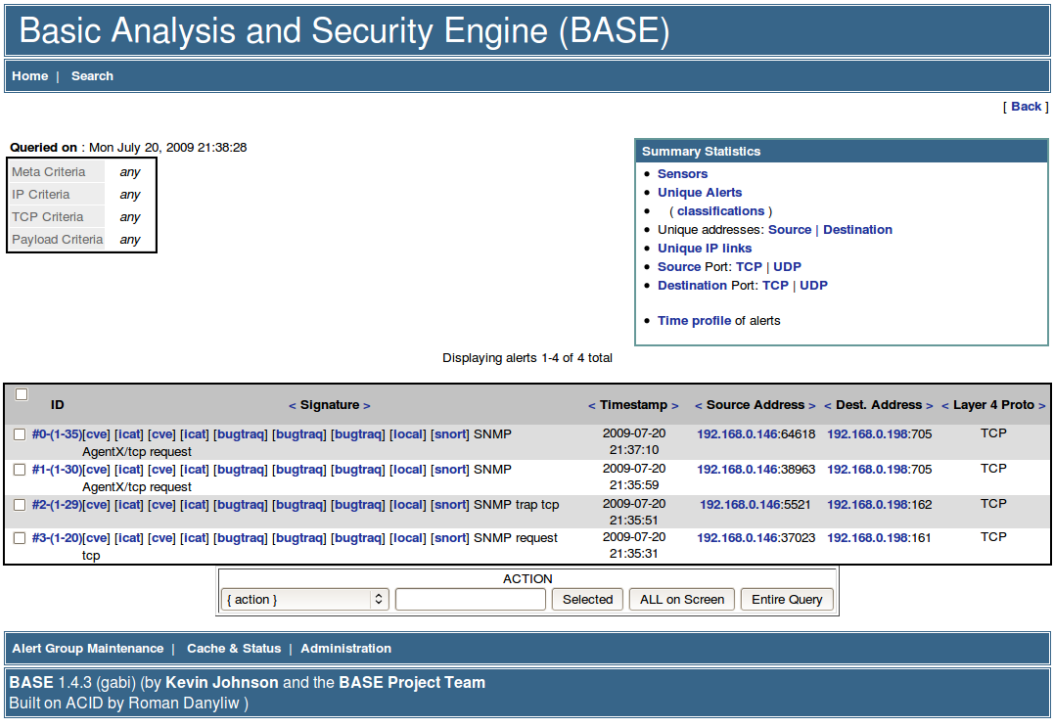


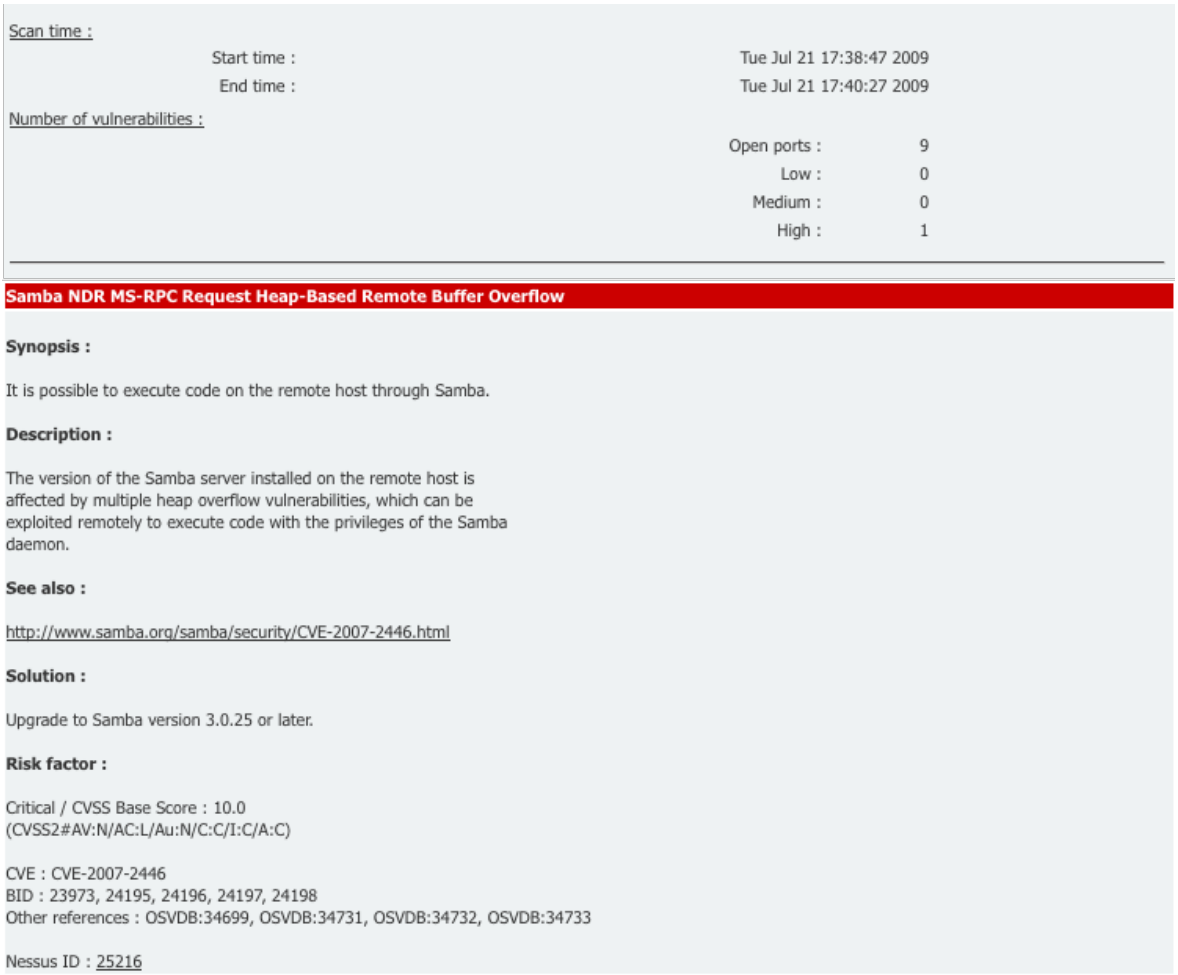
Figura 6.12: Alertas generadas por Snort en la interfaz BASE para la PoC #5

6.3.6. Vulnerabilidades en servidor Samba

El servidor Samba se encuentra basado en una implementación libre del protocolo de archivos compartidos de Microsoft Windows para sistemas de tipo UNIX. De esta forma, es posible que ordenadores con GNU/Linux, Mac OS X o Unix puedan actuar como servidores o clientes en redes Microsoft Windows [21]. Para esta prueba de concepto se ha instalado el paquete *samba-3.0.22-1ubuntu3* en el sistema objetivo y el servidor se encuentra accesible en los puertos TCP 139 y 445.

Tras una prueba inicial con todos los plugins asociados al servidor Samba en sistemas Ubuntu Linux, el escáner Nessus detecta una vulnerabilidad crítica con identificador CVE-2007-2446 asociada al plugin de Nessus 25216. La versión instalada del servidor Samba se ve afectada por varias vulnerabilidades de overflow en la memoria heap, las cuales pueden ser explotadas de forma remota para ejecutar código con el nivel privilegios de ejecución del servicio.

La figura 6.13 muestra parte del resultado una vez realizada una prueba en la que sólo se encuentra activo el plugin asociado a dichas vulnerabilidades.


Scan time :

Start time : Tue Jul 21 17:38:47 2009
End time : Tue Jul 21 17:40:27 2009

Number of vulnerabilities :

Open ports : 9
Low : 0
Medium : 0
High : 1

Samba NDR MS-RPC Request Heap-Based Remote Buffer Overflow

Synopsis :

It is possible to execute code on the remote host through Samba.

Description :

The version of the Samba server installed on the remote host is affected by multiple heap overflow vulnerabilities, which can be exploited remotely to execute code with the privileges of the Samba daemon.

See also :

http://www.samba.org/samba/security/CVE-2007-2446.html

Solution :

Upgrade to Samba version 3.0.25 or later.

Risk factor :

Critical / CVSS Base Score : 10.0
(CVSS2#AV:N/AC:L/Au:N/C:C/I:C/A:C)

CVE : CVE-2007-2446
BID : 23973, 24195, 24196, 24197, 24198
Other references : OSVDB:34699, OSVDB:34731, OSVDB:34732, OSVDB:34733

Nessus ID : 25216

Figura 6.13: Resultados del escáner Nessus en la PoC #6

La figura 6.14 muestra las alertas generadas por Snort en la interfaz BASE. Se comprueba que el IDS detecta, tan sólo, un acceso al servicio Netbios residente en el puerto 445 y no

asociado con vulnerabilidad alguna. Sin embargo, las vulnerabilidades detectadas pueden ser explotadas mediante el envío de una solicitud MS-RPC manipulada adecuadamente. El escáner Nessus accede efectivamente al sistema y realiza un análisis basado en comprobaciones de escritura con diferentes parámetros en paquetes RPC para los cuales Snort no genera ninguna alerta. Es, por tanto, un caso de falso negativo en el que un ataque puede confundirse con un acceso legitimo.

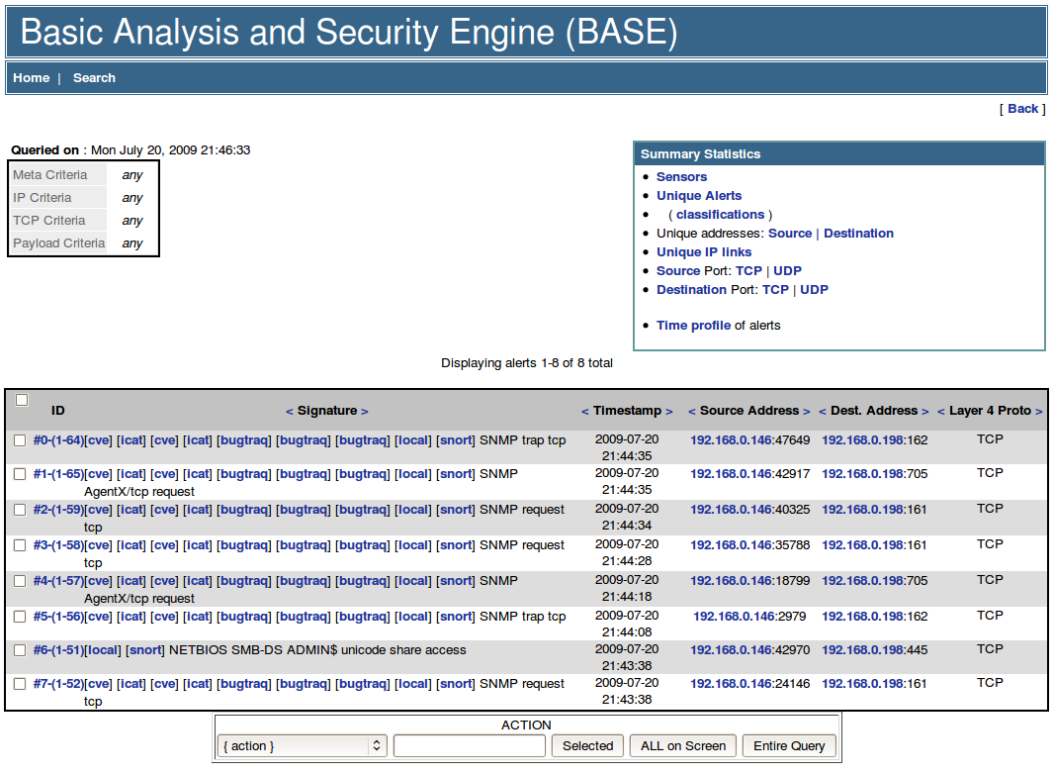


Figura 6.14: Alertas generadas por Snort en la interfaz BASE para la PoC #6

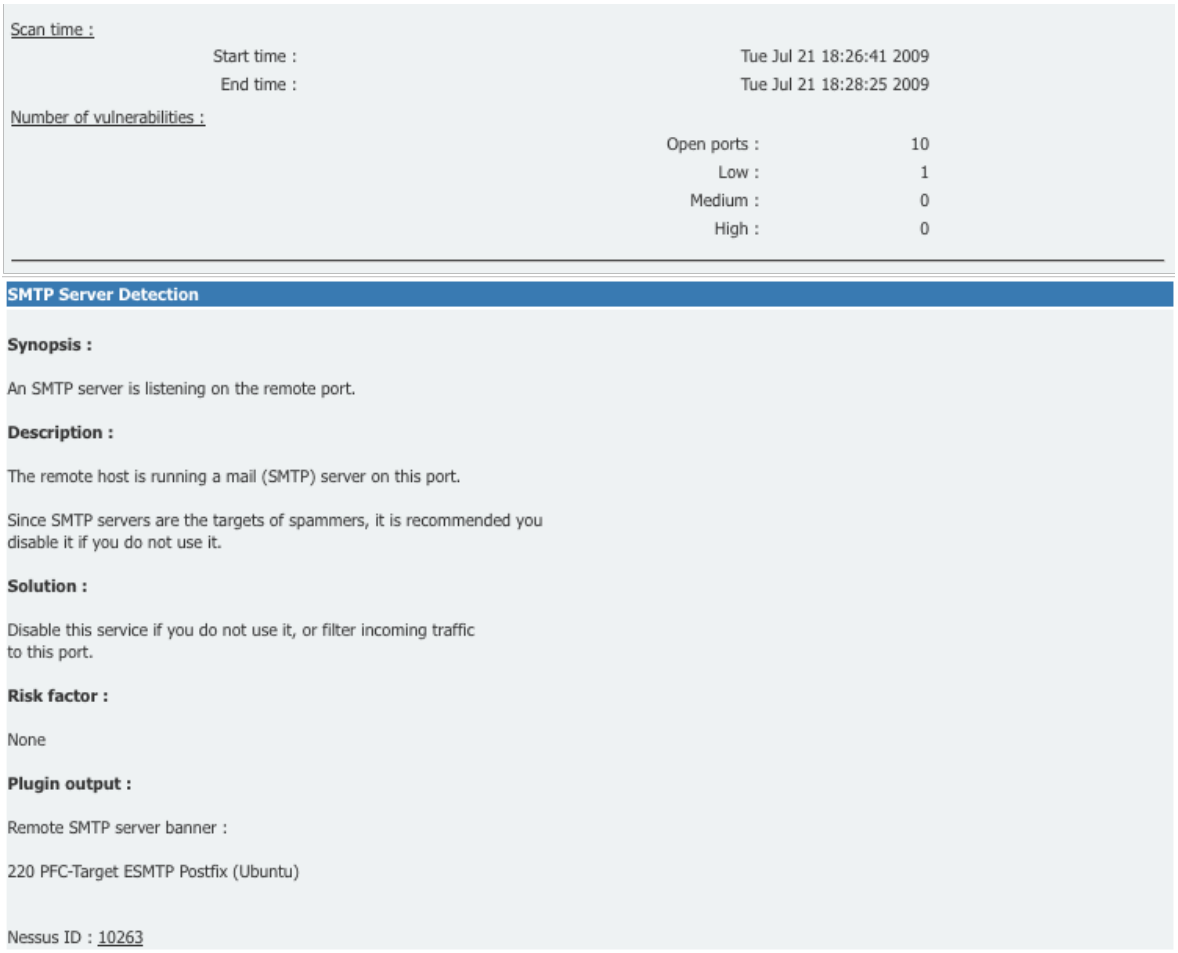
6.3.7. Vulnerabilidades en servidor SMTP

El Protocolo Simple de Transferencia de Correo *(Simple Mail Transfer Protocol, SMTP)* es un protocolo de la capa de aplicación basado en texto utilizado para el intercambio de mensajes de correo electrónico entre sistemas. Se encuentra definido en el RFC 2821 [22]. El servidor Postfix es un Agente de Transporte de Correo (MTA) de código abierto. Implementa el protocolo SMTP y permite enrutar y enviar correo electrónico [20]. Para esta prueba de concepto se ha instalado en el sistema objetivo el servidor Postfix mediante el paquete *postfix-2.2.10-1* y éste se encuentra accesible en el puerto TCP 25.

Tras una prueba inicial con todos los plugins asociados al servidor Samba en sistemas Ubuntu Linux, el escáner Nessus detecta una vulnerabilidad de perfil bajo asociada al plugin 10263. El servidor Postfix es identificado mediante la conexión y obtención del *banner* del servicio:

```
220 PFC-Target ESMTP Postfix (Ubuntu)
```

Los servidores SMTP suelen ser habitualmente objetivo de *spammers* y el escáner Nessus recomienda la desactivación del servicio en caso de que no se esté usando o la instalación de un sistema de filtrado de mensajes. La figura 6.15 muestra parte del resultado una vez realizada una prueba en la que sólo se encuentra activo el plugin asociado a esta detección.



Scan time :

Start time :	Tue Jul 21 18:26:41 2009
End time :	Tue Jul 21 18:28:25 2009

Number of vulnerabilities :

Open ports :	10
Low :	1
Medium :	0
High :	0

SMTP Server Detection

Synopsis :

An SMTP server is listening on the remote port.

Description :

The remote host is running a mail (SMTP) server on this port.

Since SMTP servers are the targets of spammers, it is recommended you disable it if you do not use it.

Solution :

Disable this service if you do not use it, or filter incoming traffic to this port.

Risk factor :

None

Plugin output :

Remote SMTP server banner :

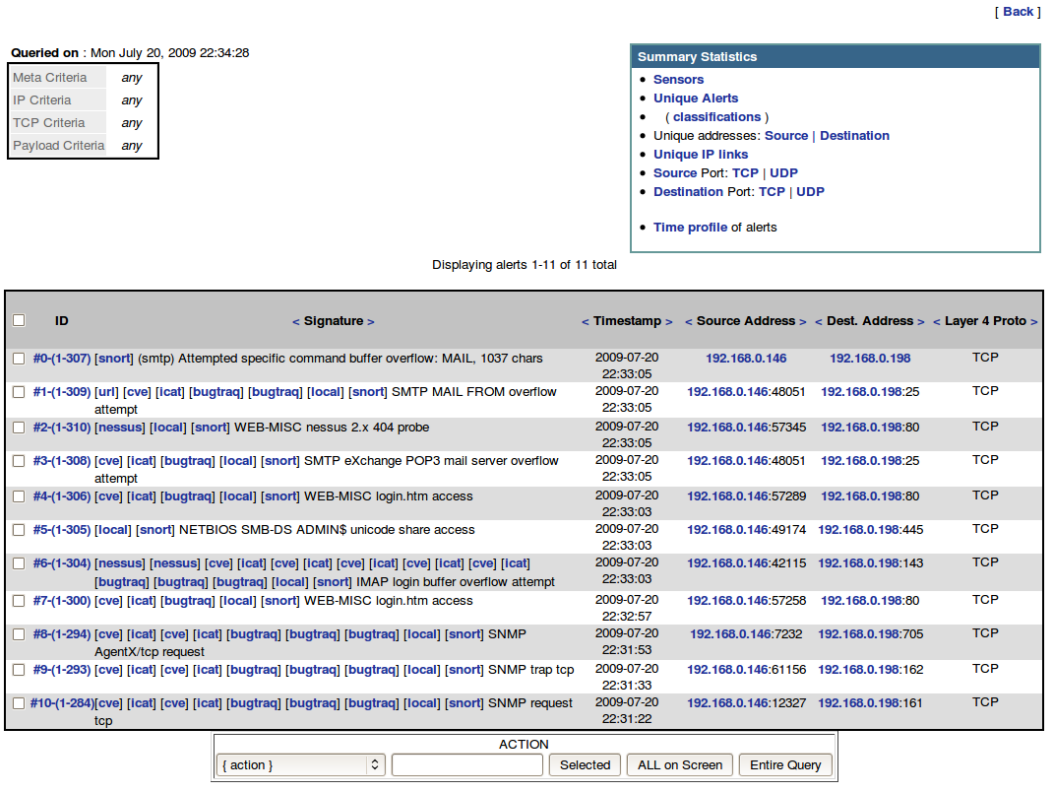
220 PFC-Target ESMTP Postfix (Ubuntu)

Nessus ID : 10263

Figura 6.15: Resultados del escáner Nessus en la PoC #7

La figura 6.16 muestra las alertas generadas por Snort en la interfaz BASE. Se comprueba

que a pesar de que Nessus sólo muestra una vulnerabilidad de perfil bajo y el único plugin activo es el que permite identificar la presencia del servidor, el IDS ha detectado varios intentos de provocar un overflow en el servidor SMTP.


[Back]

Queried on : Mon July 20, 2009 22:34:28

Meta Criteria	any
IP Criteria	any
TCP Criteria	any
Payload Criteria	any

Summary Statistics

- Sensors
- Unique Alerts
- (classifications)
- Unique addresses: Source | Destination
- Unique IP links
- Source Port: TCP | UDP
- Destination Port: TCP | UDP
- Time profile of alerts

Displaying alerts 1-11 of 11 total

ID	< Signature >	< Timestamp >	< Source Address >	< Dest. Address >	< Layer 4 Proto >
#0-(1-307)	[snort] (smtp) Attempted specific command buffer overflow: MAIL, 1037 chars	2009-07-20 22:33:05	192.168.0.146	192.168.0.198	TCP
#1-(1-309)	[url] [cve] [icat] [bugtraq] [bugtraq] [local] [snort] SMTP MAIL FROM overflow attempt	2009-07-20 22:33:05	192.168.0.146:48051	192.168.0.198:25	TCP
#2-(1-310)	[nessus] [local] [snort] WEB-MISC nessus 2.x 404 probe	2009-07-20 22:33:05	192.168.0.146:57345	192.168.0.198:80	TCP
#3-(1-308)	[cve] [icat] [bugtraq] [local] [snort] SMTP eXchange POP3 mail server overflow attempt	2009-07-20 22:33:05	192.168.0.146:48051	192.168.0.198:25	TCP
#4-(1-306)	[cve] [icat] [bugtraq] [local] [snort] WEB-MISC login.htm access	2009-07-20 22:33:03	192.168.0.146:57289	192.168.0.198:80	TCP
#5-(1-305)	[local] [snort] NETBIOS SMB-DS ADMIN$ unicode share access	2009-07-20 22:33:03	192.168.0.146:49174	192.168.0.198:445	TCP
#6-(1-304)	[nessus] [nessus] [cve] [icat] [cve] [icat] [cve] [icat] [cve] [icat] [cve] [icat] [bugtraq] [bugtraq] [bugtraq] [bugtraq] [local] [snort] IMAP login buffer overflow attempt	2009-07-20 22:33:03	192.168.0.146:42115	192.168.0.198:143	TCP
#7-(1-300)	[cve] [icat] [bugtraq] [local] [snort] WEB-MISC login.htm access	2009-07-20 22:32:57	192.168.0.146:57258	192.168.0.198:80	TCP
#8-(1-294)	[cve] [icat] [cve] [icat] [bugtraq] [bugtraq] [bugtraq] [local] [snort] SNMP AgentX/tcp request	2009-07-20 22:31:53	192.168.0.146:7232	192.168.0.198:705	TCP
#9-(1-293)	[cve] [icat] [cve] [icat] [bugtraq] [bugtraq] [bugtraq] [local] [snort] SNMP trap tcp	2009-07-20 22:31:33	192.168.0.146:61156	192.168.0.198:162	TCP
#10-(1-284)	[cve] [icat] [cve] [icat] [bugtraq] [bugtraq] [bugtraq] [local] [snort] SNMP request tcp	2009-07-20 22:31:22	192.168.0.146:12327	192.168.0.198:161	TCP

ACTION { action } Selected ALL on Screen Entire Query

Figura 6.16: Alertas generadas por Snort en la interfaz BASE para la PoC #7

Para verificar si dicho ataque se ha llevado a cabo por el escáner Nessus se he realizado una captura de tráfico mediante Wireshark. La figura 6.17 representa el intercambio de tráfico asociado a la alerta:

```
2590 || SMTP MAIL FROM overflow attempt || bugtraq,10290
|| cve,CAN-2004-0399 || url,www.guninski.com/exim1.html
```

A pesar de que la detección es correcta, el puerto desde el que se realiza el ataque es identificado por Snort como el 48051, mientras que Wireshark muestra que el puerto de origen en la máquina atacante es 56463. Se han marcado en rojo los datos enviados por el escáner para provocar el fallo.

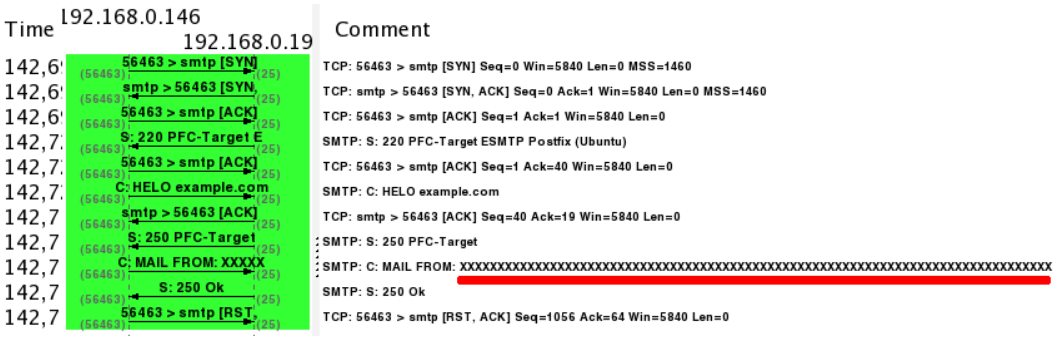


Figura 6.17: Ataque por overflow en el campo FROM en la PoC #7

La figura 6.18 representa el intercambio de tráfico asociado a las demás alertas de tipo overflow en el puerto 25 y pertenecientes todas a un mismo flujo TCP.

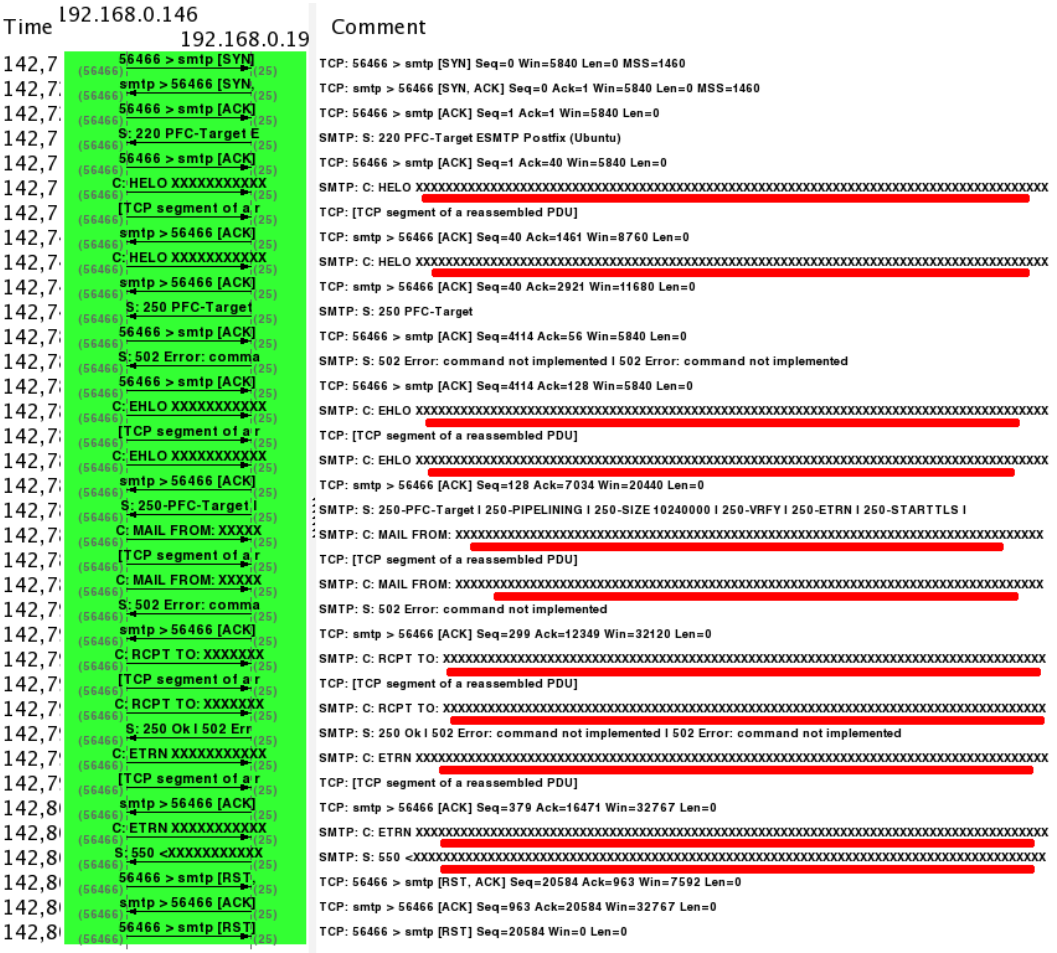


Figura 6.18: Ataques por overflow en la PoC #7

La figura 6.19 representa el intercambio de tráfico asociado a la alerta:

```
1842 || IMAP login buffer overflow attempt || bugtraq,502
|| cve,1999-0005 || cve,1999-1557 || nessus,10123 || nessus,10125
```

Se puede comprobar, a partir del diagrama de flujo, como el escáner realiza un ataque al puerto 143 e intenta provocar un overflow en el proceso de login del servicio IMAP.

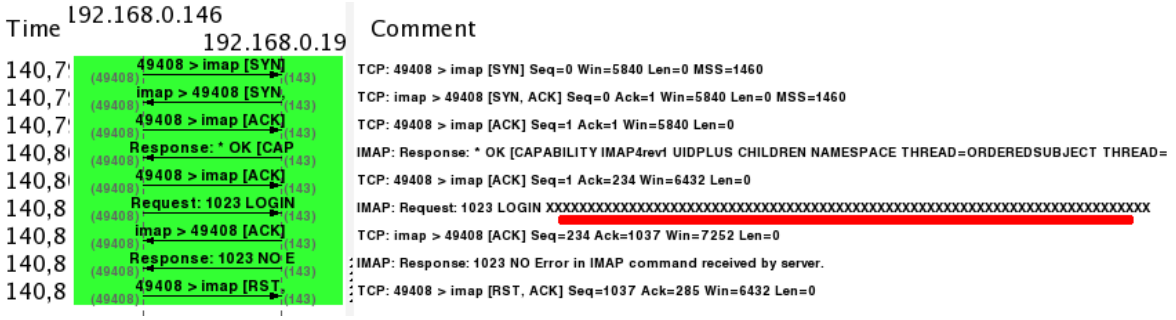


Figura 6.19: Ataque por overflow en login IMAP en la PoC #7

En esta prueba de concepto hemos comprobado que, a pesar de que el escáner Nessus detecta tan sólo la presencia del servidor, la activación de un plugin sencillo lo lleva a realizar múltiples intentos de ataque. En este caso, la respuesta del IDS Snort ha sido satisfactoria respecto a los falsos negativos. A pesar de que una revisión inicial de las alertas generadas podría indicar un gran número de falsos positivos, el análisis más profundo del tráfico mediante Wireshark ha permitido confirmar la presencia de los ataques y la correcta detección por parte de Snort.

6.4. Conclusiones

El objetivo de este capítulo y de las pruebas de concepto realizadas ha sido demostrar de forma experimental que el IDS Snort puede no detectar algunas vulnerabilidades o no proporcionar la información suficiente para identificar un ataque. Para ello, se ha utilizado el escáner de vulnerabilidades Nessus, lo que ha permitido identificar las vulnerabilidades presentes en el sistema para cada servidor y verificar la respuesta de Snort ante tales comprobaciones. Aunque el uso de un VDS es una práctica común para verificar la configuración de un IDS, las pruebas realizadas y los resultados obtenidos permiten obtener algunas conclusiones adicionales tanto sobre el funcionamiento del escáner como de la respuesta de Snort.

En primer lugar y cuando el objetivo es determinar la presencia de falsos positivos y negativos, las comprobaciones iniciales y la detección de puertos realizadas por el escáner Nessus antes de transmitir los paquetes propios de cada plugin generan mucho ruido en el IDS y dificultan la correlación entre ataques y alertas. En las pruebas realizadas, se han observado un gran número de paquetes de tipo SNMP, que si bien no representan una amenaza para el sistema, son detectados por Snort y dificultan conocer la respuesta especifica del IDS ante un plugin determinado. Por todo esto, una herramienta que pudiera realizar una comprobación de la vulnerabilidad con un número mínimo de paquetes facilitaría la identificación de la alerta generada en relación con el ataque. La solución utilizada, en estas pruebas, ha sido la verificación mediante el sniffer Wireshark del tráfico real que el escáner está enviando al objetivo. Ésto ha permitido localizar los paquetes susceptibles de ser detectados por las reglas de Snort. No obstante y aunque el VDS es una herramienta útil para conocer el estado general del IDS, se observa que si la intención es verificar la respuesta ante una vulnerabilidad específica, Nessus puede no ser suficiente o no ser la herramienta más adecuada. En muchas ocasiones la detección de la vulnerabilidad se realiza mediante la comparación de versiones y aunque esto representa un buen método para conocer la seguridad de un sistema, no permite conocer la respuesta del IDS ante el ataque real mediante exploits. La conclusión directa de esto es que la mejor forma de comprobar la respuesta del IDS ante ataques reales es realizando ataque reales, es decir, enviando paquetes que contienen código malicioso y pueden vulnerar realmente los servicios disponibles. Para un caso real de una red bajo un proceso de auditoría, sería, por tanto, recomendable la realización de un escáner mediante Nessus y posteriormente, llevar a cabo una serie de ataques a esos servicios para verificar la respuesta de Snort.

Por otra parte, se ha observado que la falta de contexto de aplicación en el IDS puede generar falsos positivos e incluso falsos negativos. En entornos dinámicos, el IDS puede estar generando alertas para servicios que no se encuentran disponibles o que a pesar de encontrarse activos pueden pertenecer a aplicación diferentes. Como se ha visto en el caso del servidor FTP, programas distintos pueden activar este servicio y mientras que un servidor puede ser vulnerable a un overflow en el login el otro puede no serlo. El IDS, sin embargo, no posee conocimiento sobre la aplicación detrás del servicio, produciéndose así falsos positivos. En una red donde conviven ambos tipos de servicios, la alerta no podría ser desactivada a riesgo de producirse falsos negativos y la explotación de la vulnerabilidad.

Capítulo 7

Detección de Exploits

7.1. Introducción

En los capítulos 5 y 6 se ha intentado analizar la respuesta del IDS Snort frente a diferentes ataques de reconocimiento y frente a un escáner de vulnerabilidades. Sin embargo, el objetivo final del motor de reglas de Snort es la identificación de código malicioso contenido en paquetes transmitidos a través de la red.

Los *exploits* son fragmentos de código o secuencias de comandos que permiten automatizar el aprovechamiento de un error, fallo o vulnerabilidad, a fin de causar un comportamiento no deseado o imprevisto en un sistema. Con frecuencia, esto incluye consecuencias tales como la toma de control del sistema, permitir una escalada de privilegios o provocar un ataque de denegación de servicio. Los exploits pueden ser escritos empleando diversos lenguajes de programación, aunque la mayoría se crean en lenguaje C y el ataque realizado por el código contenido en el exploit puede ser de diferentes tipos tales como desbordamiento de búfer, cross site scripting, format strings, injección SQL, etc [18]. Aunque el código contenido en un exploit puede ser ejecutado en un sistema local, el acceso físico al mismo es una barrera considerable a salvar por un atacante, sin embargo, cuando el código puede ejecutarse en el sistema tras recibir el exploit enviado de forma remota, el número de potenciales atacantes se dispara y el sistema detector de intrusión se vuelve imprescindible.

Si se tiene en cuenta el proceso de desarrollo tradicional de un exploit o se analiza el código que muchos exploits contienen, se podrá comprobar que una parte significativa del código es reutilizable. Por ejemplo, la mayoría de los exploits que intentan provocar un buffer

overflow tendrán que crear un buffer con *shellcode* y todos los exploits remotos utilizarán una función para acceder al socket y lanzar el ataque hacia el objetivo a través de la red. Por ello, la mayoría de los desarrolladores de exploits manejan librerías de funciones que se usan de forma habitual y pueden ser incluidas en uno y otro exploit.

El entorno Metasploit significa un paso más en el uso de librerías de funciones. Éste permite a los investigadores utilizar, no sólo librerías de código para diferentes tareas como llamadas a funciones escritas en ensamblador, funciones RPC o gestión de los buffers de memoria, sino también un motor que convierte el uso de exploits en una tarea modular, para la cual sólo es necesario introducir los parámetros deseados de forma dinámica en tiempo de ejecución. Típicamente, los exploits han sido fragmentos de código muy estáticos y Metasploit supone un avance ya que aunque cada código está diseñado para funcionar con unas instrucciones particulares, en una versión específica de un servicio que funciona sobre una versión específica de un sistema operativo, la modularidad y simplicidad del entorno facilita el proceso de desarrollo de los exploits, disminuyendo el tiempo necesario para crearlo a partir de un gran repositorio de código estable y probado [14].

Metasploit se ha convertido rápidamente en una herramienta esencial en el mundo de la seguridad práctica y aunque representa una gran ayuda para investigadores y auditores, también es una herramienta muy sencilla de utilizar por medio del motor de ejecución si tan sólo se busca lanzar un ataque. De ahí que se esté convirtiendo en un estándar para pruebas de concepto y un arma de explotación al alcance de cualquier atacante sin conocimientos avanzados.

El objetivo de este capítulo es comprobar, mediante el uso del entorno Metasploit, la respuesta del IDS Snort ante distintos ataques con exploits reales dirigidos a servicios disponibles en sistemas linux.

7.2. El Framework Metasploit

El framework Metasploit es una plataforma *open source* diseñada para facilitar el desarrollo, la comprobación y el uso de exploits con el objetivo de realizar pruebas de intrusión, desarrollo de *shellcode* y análisis de vulnerabilidades. El framework puede correr sobre prácticamente cualquier sistema basado en Unix que incluya una versión moderna del intérprete de Ruby (1.8.4+) y antes de ser liberada cada nueva versión estable, su correcto funcionamiento

es comprobado sobre las tres principales plataformas: Linux 2.6 (x86, ppc), Windows NT (2000, XP, 2003, Vista), MacOS X 10.4 (x86, ppc) y 10.5 (x86) [12].

El modelo extensible, a través del cual pueden integrarse en un solo software exploits, *payloads*, módulos NOP y codificadores, ha hecho de ésta una herramienta a la vanguardia del análisis de vulnerabilidades. Existen otras herramientas de caracteristicas similares aunque disponibles a elevados costes y bajo licencias comerciales. Metasploit, bajo licencia BSD en su nueva versión 3.2, se encuentra disponible de forma gratuita y ha supuesto una revolución al permitir a desarrolladores disponer de un entorno de gran potencia para la investigación en seguridad [7].

7.2.1. Instalación y Funcionamiento

Para llevar a cabo la instalación del framework en un sistema linux tan sólo es necesario descomprimir el paquete, acceder al directorio creado y ejecutar la interfaz deseada. Existen cuatro opciones: la interfaz de consola, la interfaz gráfica, la linea de comandos y la interfaz web.

La consola

Se puede iniciar mediante el comando `msfconsole`, tras lo cual aparecerá un símbolo de sistema. Este es el modo principal y escribiendo `show exploits`, se listarán todos los exploits disponibles. El comando `help` presentará una lista de los comandos válidos. La interfaz de consola es la más rapida y versatil de todas y permite acceder a comandos del sistema sobre el que se está ejecutando MS.

La interfaz gráfica

Se inicia mediante el comando `msfgui` y permite realizar las mismas operaciones que desde la consola desde el entorno gráfico.

La linea de comandos

Si se desea automatizar la prueba de una serie de exploits o simplemente no se quiere usar una interfaz interactiva, es posible invocar la linea de comandos mediante `msfcli`. En este modo, el módulo a probar y sus correspondientes opciones son pasadas como parámetros.

La interfaz web

La interfaz web está basada en *Ruby on Rails* y su servidor puede lanzarse ejecutando `msfweb`. Por defecto, éste escuchará en la dirección loopback en el puerto 55555. La interfaz principal consiste en una barra de tareas desde la que son accesibles los distintos módulos para configuración y ejecución de exploits o payloads como se muestra en la figura 7.1

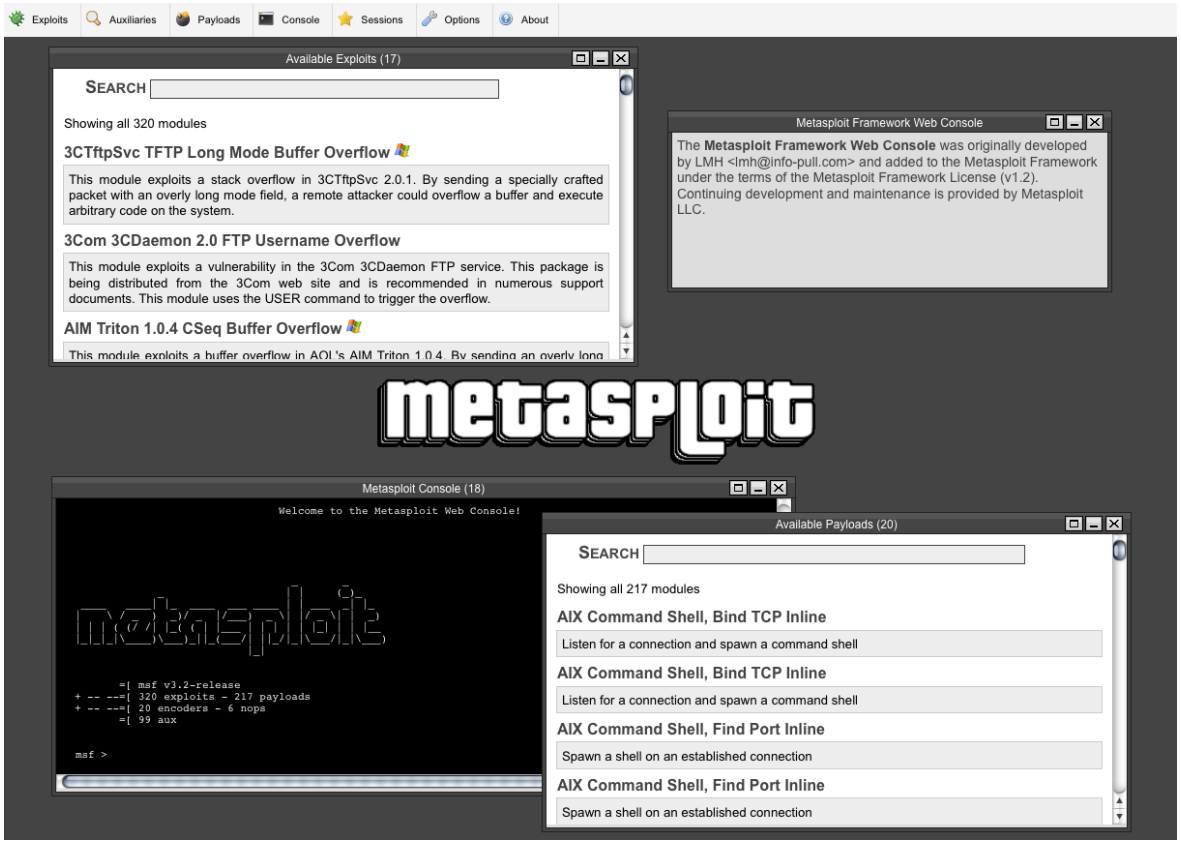


Figura 7.1: Interfaz web del Framework Metasploit.

7.2.2. Actualizaciones

El framework completo puede ser actualizado utilizando un cliente *Subversion*. Para ello es necesario cambiar al directorio de instalación y ejecutar `svn update`. El sitio *metasploit.com* debe ser el primer lugar a la hora de comprobar si existen actualizaciones de los módulos o si una nueva versión ha sido liberada. En dicha página también se puede encontrar una base de datos de códigos de operación (*opcodes*) y *shellcodes*

7.3. Pruebas de concepto

Al igual que en el capítulo 6, las reglas utilizadas para llevar a cabo las siguientes pruebas son las publicadas el 16 de Mayo de 2009 para usuarios con suscripción y el 16 de Junio de 2009 para usuarios solamente registrados. En este caso, se han utilizado los mismos servicios para el ataque a los servidores IMAP y Samba, se han instalado dos nuevos servidores PeerCast y Squid y se ha cambiado la máquina objetivo 192.168.0.199 por el IDS para probar un ataque directo a una vulnerabilidad en Snort.

La figura 7.2 muestra la consola de Metasploit y el listado de exploits disponibles para sistemas Linux. Ya que el número de exploits disponibles por defecto para sistemas Linux es bajo en comparación con otros sistemas, durante el desarrollo de este proyecto se han intentado probar todos ellos para verificar la respuesta del IDS. Sin embargo, algunos exploits están específicamente diseñados para software propietario del cual no se ha podido disponer para instalar en el sistema objetivo. Finalmente, se han obtenido resultados tanto en Metasploit como en Snort para cinco de ellos y aunque se han probado otros exploits para software libre disponible como Postfix o MySQL, Metasploit no ha sido capaz de conseguir una conexión con el servicio sobre la que lanzar el ataque. Para los casos de prueba que se muestran, Metasploit ha conseguido conectar con el servidor y enviar el exploit, permitiendo así a Snort analizar el tráfico en busca de código malicioso.

```
[*] Searching loaded modules for pattern 'linux'...

Exploits
========

   Name                                  Description
   ----                                  -----------
   linux/games/ut2004_secure             Unreal Tournament 2004 "secure" Overflow (Linux)
   linux/http/gpsd_format_string         Berlios GPSD Format String Vulnerability
   linux/http/linksys_apply_cgi          Linksys apply.cgi buffer overflow
   linux/http/peercast_url               PeerCast <= 0.1216 URL Handling Buffer Overflow (linux)
   linux/ids/snortbopre                  Snort Back Orifice Pre-Preprocessor Remote Exploit
   linux/imap/imap_uw_lsub               UoW IMAP server LSUB Buffer Overflow
   linux/madwifi/madwifi_giwscan_cb      Madwifi SIOCGIWSCAN Buffer Overflow
   linux/misc/gld_postfix                GLD (Greylisting Daemon) Postfix Buffer Overflow
   linux/misc/ib_inet_connect            Borland InterBase INET_connect() Buffer Overflow
   linux/misc/ib_jrd8_create_database    Borland InterBase jrd8_create_database() Buffer Overflow
   linux/misc/ib_open_marker_file        Borland InterBase open_marker_file() Buffer Overflow
   linux/misc/ib_pwd_db_aliased          Borland InterBase PWD_db_aliased() Buffer Overflow
   linux/mysql/mysql_yassl               MySQL yaSSL SSL Hello Message Buffer Overflow
   linux/pptp/poptop_negative_read       Poptop Negative Read Overflow
   linux/proxy/squid_ntlm_authenticate   Squid NTLM Authenticate Overflow
   linux/samba/lsa_transnames_heap       Samba lsa_io_trans_names Heap Overflow
```

Figura 7.2: Exploits para Linux presentes por defecto en MSF.

7.3.1. Explotación de servidor IMAP

Para esta prueba de concepto, se va a utilizar el módulo *UoW IMAP server LSUB Buffer Overflow*, el cual permite explotar un buffer overflow en el comando LSUB del servidor IMAP implementado por la Universidad de Washington (UoW IMAP). Esta vulnerabilidad sólo puede ser explotada si el atacante dispone de un login de usuario y un password válidos.

El servidor IMAP no verifica la longitud de los argumentos que se pasan a los comandos LIST, COPY, RENAME, FIND y LUSB, lo cual puede ocasionar un desbordamiento de buffer. Si se envía un paquete especialmente formado, un atacante puede conseguir ejecutar código arbitrario en la máquina que alberga al servidor con los mismos privilegios del usuario, ocasionando una pérdida de integridad.

La figura 7.3 muestra la consola de Metasploit y la salida de la ejecución del módulo seleccionado. Se comprueba como el módulo establece una conexión con el servicio, se identifica de forma correcta y envía el exploit para provocar el overflow. Aunque se consigue enviar el código, éste no consigue explotar la vulnerabilidad en el servicio.

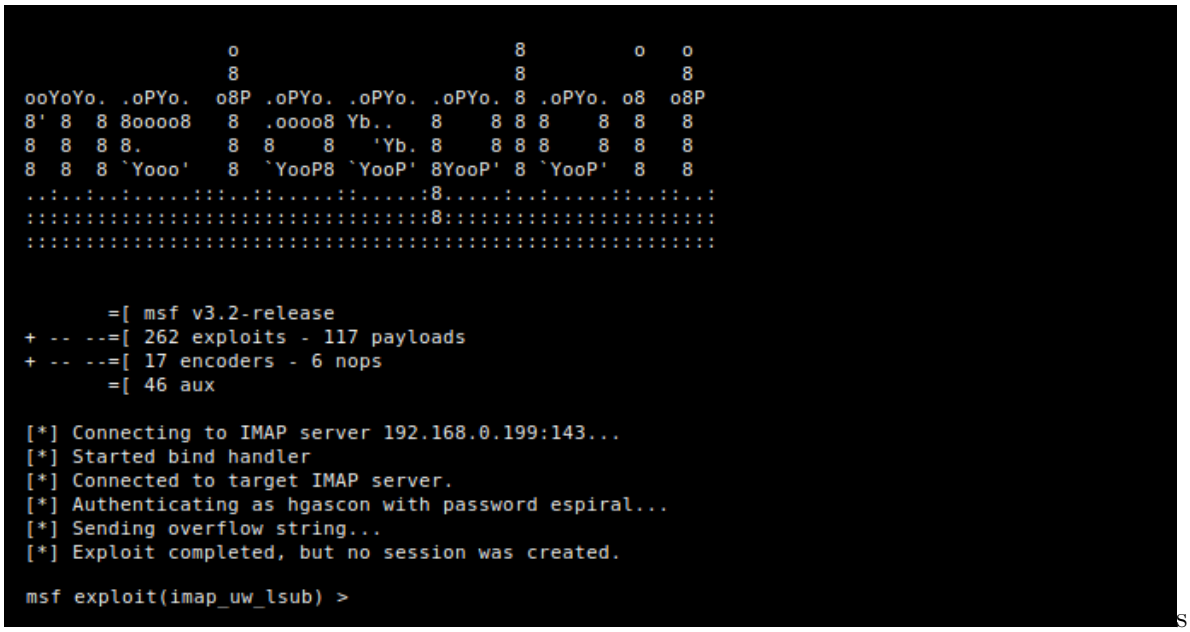


Figura 7.3: Consola de resultados en MSF en la PoC #1.

La figura 7.4 muestra las alertas generadas por Snort en la interfaz BASE. Se comprueba como el IDS detecta correctamente el código del exploit en el paquete enviado. A diferencia

de otros casos mostrados anteriormente, no se producen falsos positivos o negativos sino una identificación clara del ataque y se genera su alerta correspondiente.

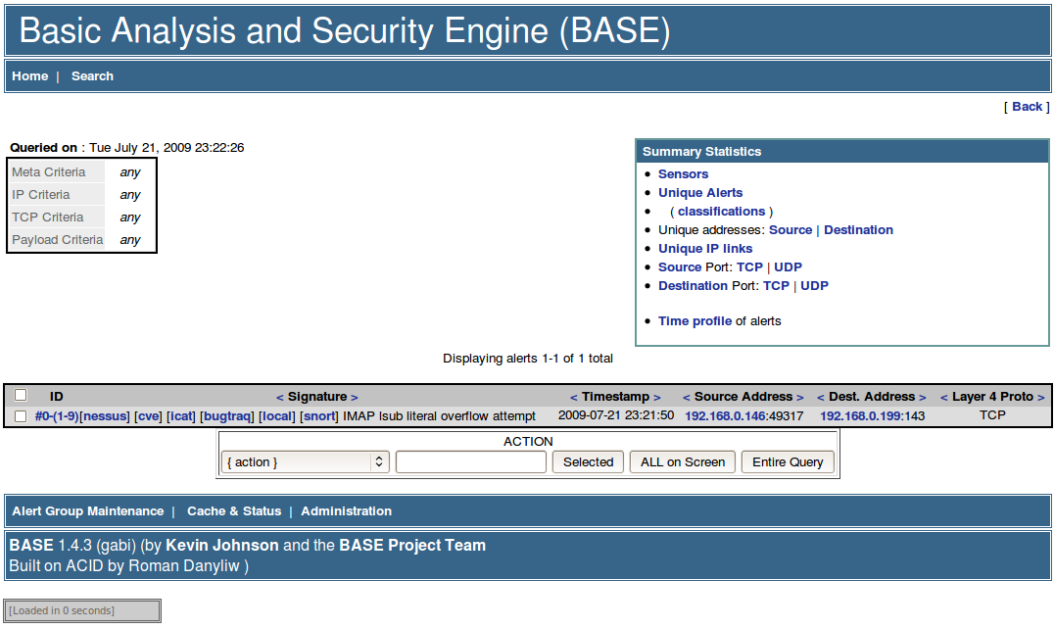


Figura 7.4: Alertas generadas en BASE en la PoC #1.

La figura 7.5 muestra el intercambio de tramas entre el atacante y el objetivo. El intento de provocar el buffer overflow aparece marcado en rojo.

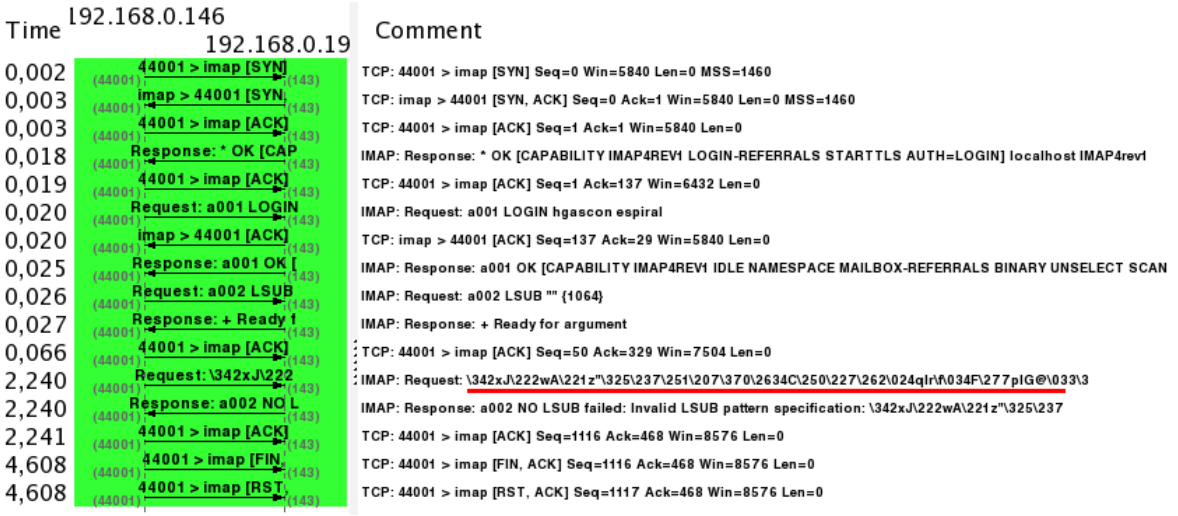


Figura 7.5: Flujo de tráfico observado mediante Wireshark en la PoC #1.

7.3.2. Explotación de servidor Peercast

Para esta prueba de concepto, se va a utilizar el módulo *PeerCast <= 0.1216 URL Handling Buffer Overflow (linux)*, el cual permite explotar un stack overflow en versiones inferiores a la 0.1216 del servidor PeerCast. Éste es un servicio de difusión de audio open source que utiliza tecnología P2P y permite a cualquier usuario instalar tanto un cliente como un servidor de emphstreaming.

El módulo utilizado en esta prueba de concepto permite explotar una vulnerabilidad causada por un error en el manejo de los parámetros de la URL. La función procConnectArgs() no verifica la longitud de los parámetros que se pasan a través de la URL, lo cual puede ocasionar un desbordamiento de pila. Si se envía una URL especialmente formada, un atacante puede conseguir ejecutar código arbitrario en la máquina que alberga al servidor con los mismos privilegios del usuario.

La figura 7.6 muestra la consola de Metasploit y la salida de la ejecución del módulo seleccionado. En este caso no es necesaria una conexión con el servicio, tan sólo intentar acceder al servicio por medio de la URL que contiene los parámetros adecuados para provocar el overflow. Aunque el acceso se realiza correctamente, el ataque no consigue explotar la vulnerabilidad en el servicio.

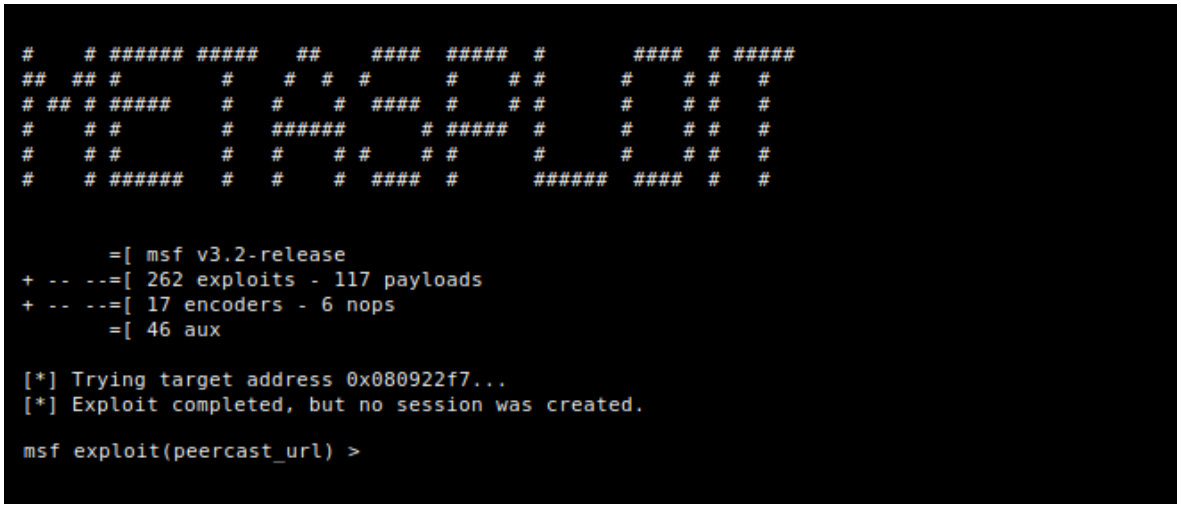


Figura 7.6: Consola de resultados en MSF en la PoC #2.

La figura 7.7 muestra las alertas generadas por Snort en la interfaz BASE. Se comprueba

que a pesar de que se han generado varias alertas por paquetes ICMP que no representan una amenaza y pueden ser consideradas falsos positivos, el ataque es detectado correctamente.

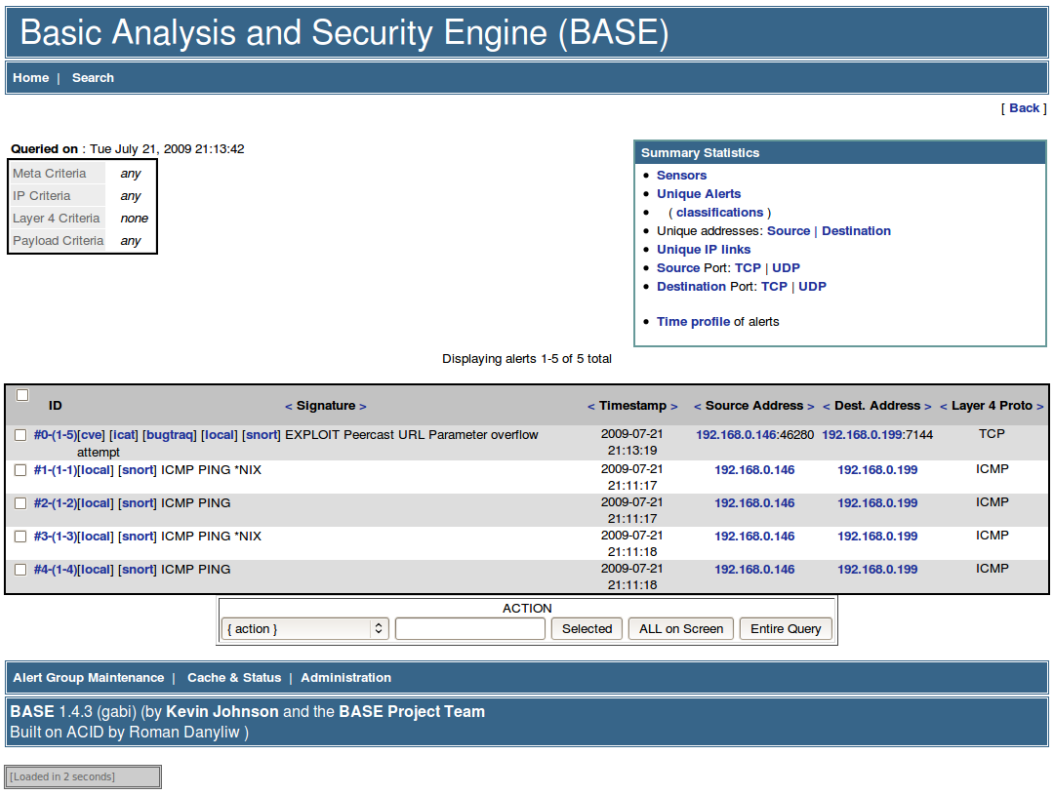


Figura 7.7: Alertas generadas en BASE en la PoC #2.

La figura 7.8 muestra el intercambio de tramas entre el atacante y el objetivo. El intento de provocar el overflow aparece marcado en rojo.

Figura 7.8: Flujo de tráfico observado mediante Wireshark en la PoC #2.

7.3.3. Explotación de servidor Samba

Para esta prueba de concepto, se va a utilizar el módulo *Samba lsa_io_trans_names Heap Overflow*, el cual permite explotar un heap overflow en el servicio LSA RPC del servidor Samba. Este módulo utiliza el método de sobreescritura de fragmentos TALLOC (talloc chunk method, Ramon and Adriano), que sólo funciona en las versiones de Samba de 3.0.21 a 3.0.24.

La aplicación falla al comprobar los argumentos que se pasan en las solicitudes RPC a la interfaz SPOOLS RPC, lo cual puede ocasionar un desbordamiento de heap. Si se envía una solicitud especialmente formada a RFNPCNEX, un atacante puede conseguir sobreescribir el espacio de heap y ejecutar código arbitrario en la máquina que alberga al servidor con los mismos privilegios del usuario.

La figura 7.9 muestra la consola de Metasploit y la salida de la ejecución del módulo seleccionado. Se comprueba como el módulo establece una conexión con el servicio y consigue llamar a la función vulnerable para envíar el exploit y provocar el overflow. Aunque se consigue enviar el código, éste no consigue explotar la vulnerabilidad en el servicio. En este caso, la versión instalada del servidor es la 3.0.22, por lo que debería ser vulnerable, sin embargo, este es un ataque basado en fuerza bruta y las condiciones especificas para que el exploit tenga éxito son reducidas.

La figura 7.10 muestra las alertas generadas por Snort en la interfaz BASE. En este caso, Snort está generando dos tipos de alerta. La primera, corresponde con una identificación del código contenido en el *shellcode* de los exploits y la segunda se corresponde con una detección del propio exploit en las solcitudes RPC. No se produce por tanto ningun falso negativo o positivo y la identificación del ataque, además de ser correcta, se realiza por dos medios diferentes.

7.3.4. Explotación de Snort Back Orifice

Para esta prueba de concepto, se va a utilizar el módulo *Snort Back Orifice Pre-Preprocessor Remote Exploit*, el cual permite explotar un stack overflow en el módulo del preprocesador Back-Orifice incluido en las versiones de Snort 2.4.0, 2.4.1, 2.4.2, y 2.4.3. Para este proyecto, se está trabajando con la versión 2.8.4.1, que aunque no ha de ser vulnerable, nos permitirá conocer la respuesta del IDS ante un ataque directo a la máquina que lo alberga, lo cual supone un caso especial de ataque cuyo análisis plantea un caso diferente e interesante.

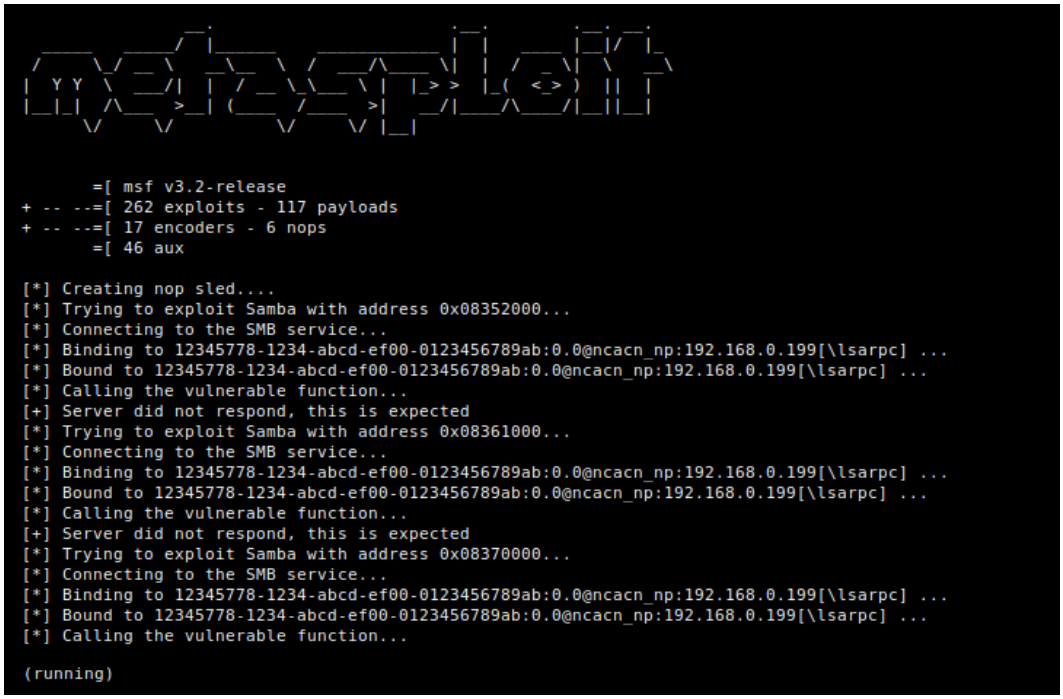


Figura 7.9: Consola de resultados en MSF en la PoC #2.

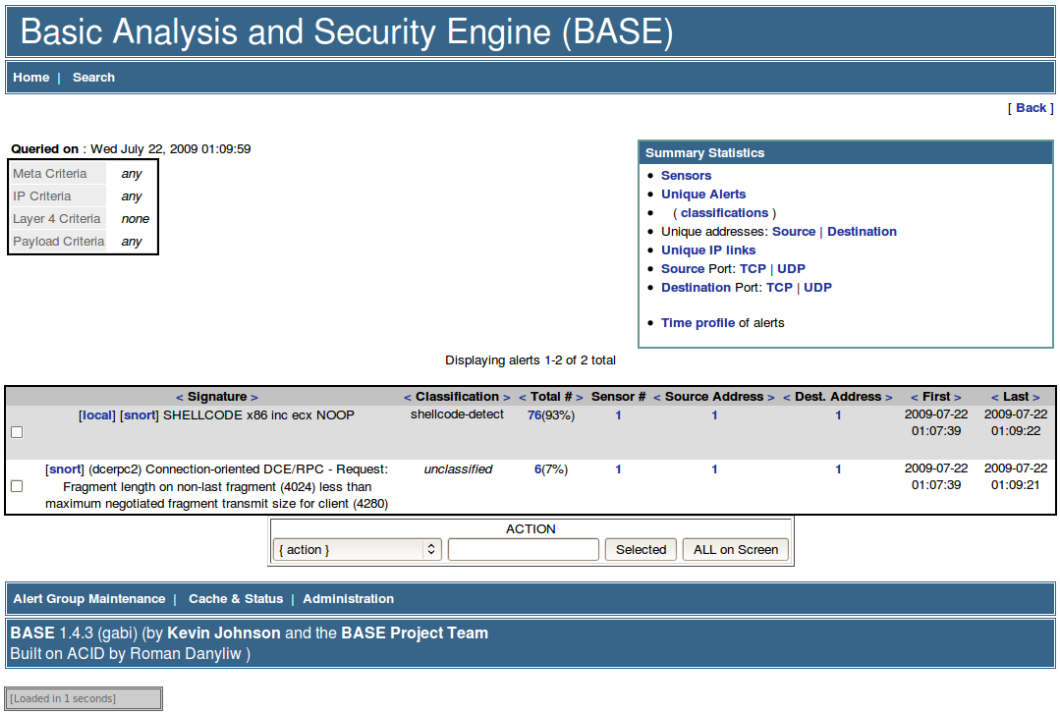


Figura 7.10: Alertas generadas en BASE en la PoC #3.

Esta vulnerabilidad está causada por un fallo en el preprocesador Back Orifice a la hora de validar paquetes de tipo UDP, lo cual puede ocasionar un desbordamiento de la pila y comprometer completamente el sensor IDS, dando a un atacante privilegios completos de administración de la máquina y del IDS. Para ello, sería necesario tan sólo enviar un paquete UDP especialmente formado y el atacante podría conseguir ejecutar código arbitrario en la máquina y obtener el control de la misma.

La figura 7.11 muestra la consola de Metasploit y la salida de la ejecución del módulo seleccionado. En este caso, el módulo no necesita establecer una conexión con la máquina objetivo, tan sólo enviar el paquete que contiene el exploit para intentar vulnerar el servicio. Aunque se consigue enviar el código, como se esperaba, éste no consigue explotar la vulnerabilidad en el IDS.

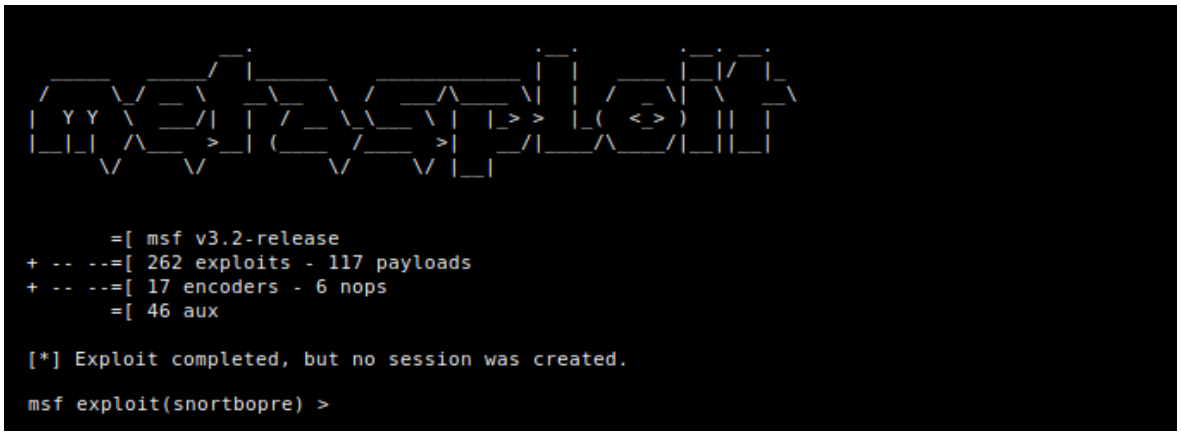


Figura 7.11: Consola de resultados en MSF en la PoC #4.

La figura 7.12 muestra las alertas generadas por Snort en la interfaz BASE. Como era de esperar, puesto que se trata de una vulnerabilidad que afecta propiamente al sistema Snort, se ha realizado una identificación correcta del ataque.

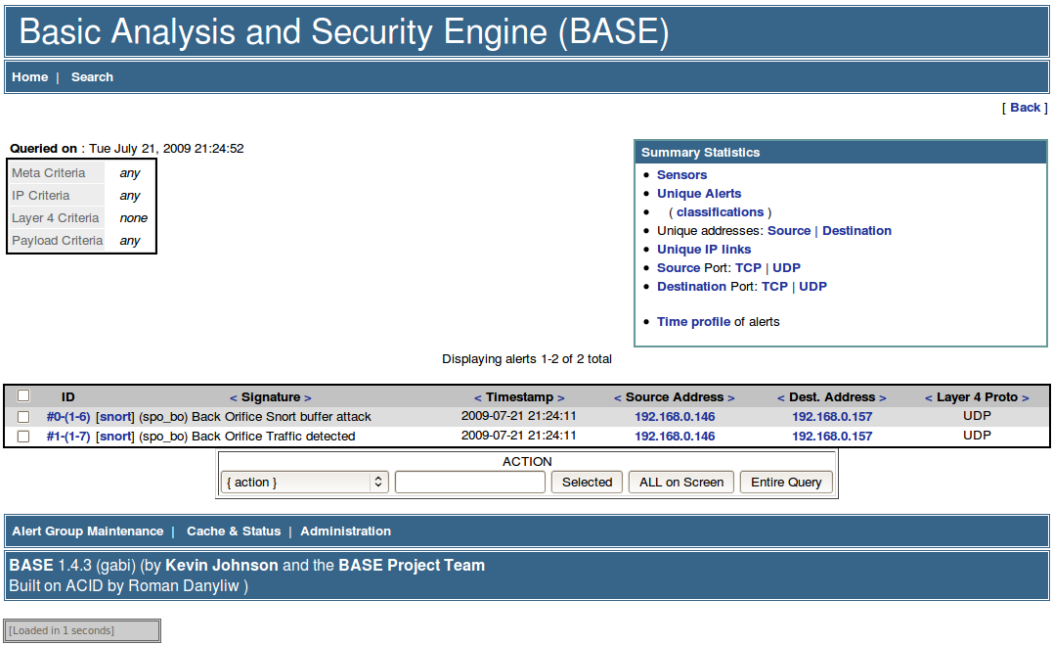


Figura 7.12: Alertas generadas en BASE en la PoC #4.

La figura 7.13 muestra el intercambio de tramas entre el atacante y el objetivo. En este caso, el ataque se limita al envío de un paquete UDP adecuadamente formado que es detectado por la máquina de Snort. El puerto seleccionado por el exploit no es determinante ya que el sensor IDS analiza todo el tráfico dirigido a su interfaz o a través de ella. La implementación de la pila TCP/IP de la máquina donde reside el IDS, responde con un paquete de error ICMP indicando que el puerto al que se ha enviado la solicitud se encuentra cerrado.

Figura 7.13: Flujo de tráfico observado mediante Wireshark en la PoC #4.

7.3.5. Explotación de servidor Squid

Squid es un popular programa de software libre que implementa un servidor proxy y un servidor para caché de páginas web. Permite acelerar el acceso a un servidor web, guardando en caché peticiones repetidas a DNS y otras búsquedas para un grupo de usuarios que comparte recursos. También permite añadir seguridad mediante filtrado de tráfico y está especialmente diseñado para ejecutarse bajo entornos tipo Unix [23].

Para esta prueba de concepto, se va a utilizar el módulo *Squid NTLM Authenticate Overflow*, el cual permite explotar un buffer overflow en la función de autenticación NTLM de Squid. Debido a una comprobación errónea en la función ntlm_check_auth(), es posible provocar un desbordamiento de pila al introducir el valor del hash de LanMan, formato utilizado por sistemas antiguos Microsoft Windows para almacenar passwords con una longitud menor de 15 caracteres. Si se envía una solicitud especialmente formada, un atacante podría conseguir ejecutar código arbitrario en el sistema con los mismos privilegios bajo los que se esta ejecutando Squid. Esta vulnerabilidad sólo puede ser explotada si Squid ha sido compilado con soporte para autenticación NTLM.

La figura 7.14 muestra la consola de Metasploit y la salida de la ejecución del módulo seleccionado. Se comprueba como el módulo establece una conexión con el servicio, comienza la negociación del proceso de autenticación NTLM y envía el paquete especialmente formado para provocar el fallo del sistema. Al tratarse de un ataque por fuerza bruta, el proceso de repite de forma iterativa y aunque no se consigue explotar la vulnerabilidad en el servicio un número mínimo de intentos nos permite comprobar la respuesta del IDS.

La figura 7.15 muestra las alertas generadas por Snort en la interfaz BASE. Se comprueba como el IDS detecta correctamente el código del exploit en cada una de las peticiones enviadas por Metasploit. No se produce ningún caso de falso positivo o negativo ya que cada paquete que contiene el exploit es identificado correctamente y de forma individual, generándose su alerta correspondiente.

La figura 7.16 muestra el intercambio de tramas entre el atacante y el objetivo. En este diagrama de flujo todos los paquetes pertenecen a un mismo stream TCP en el que se realizan dos llamadas a la función de autenticación con el objetivo de provocar el fallo y aprovechar la vulnerabilidad.

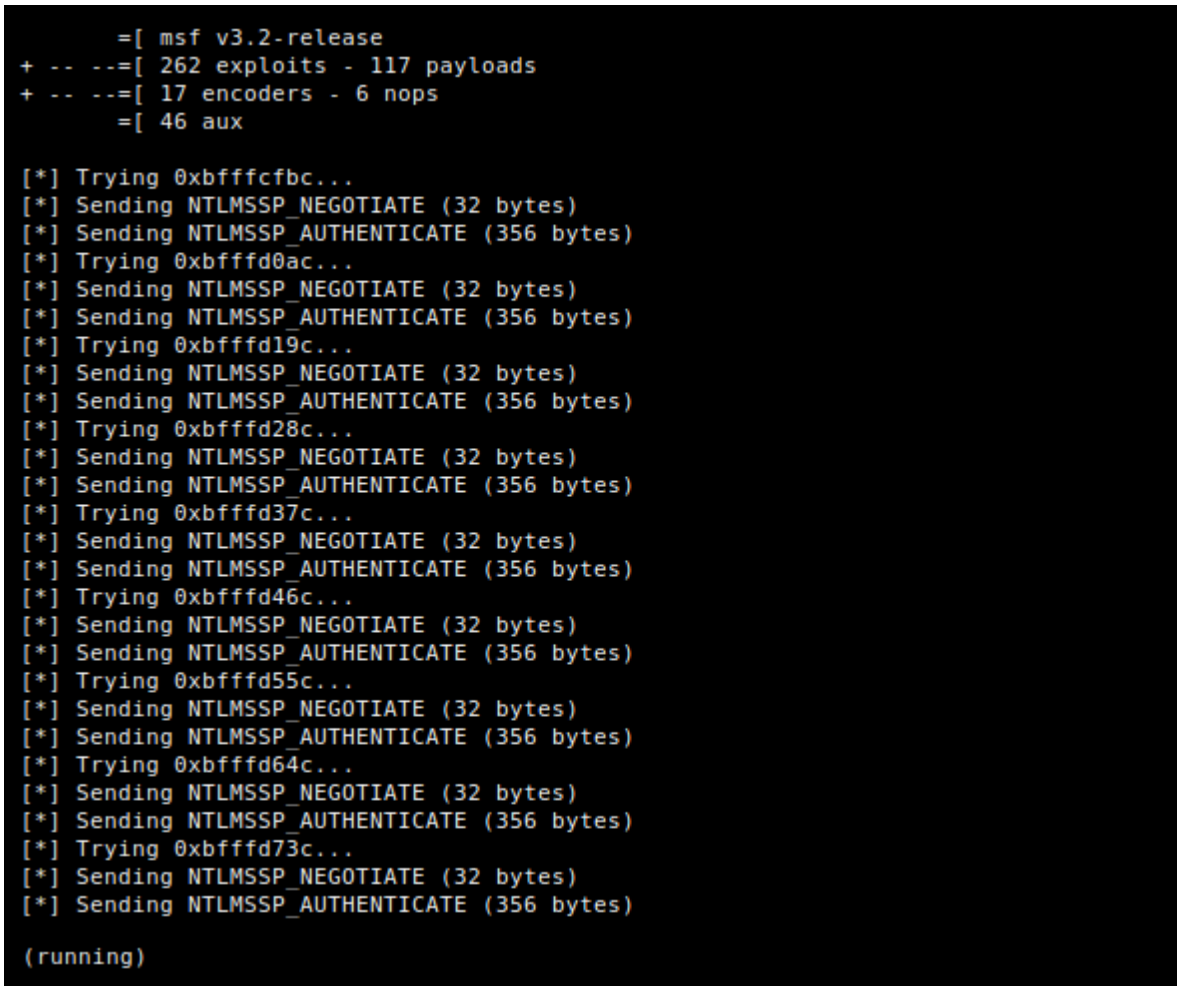
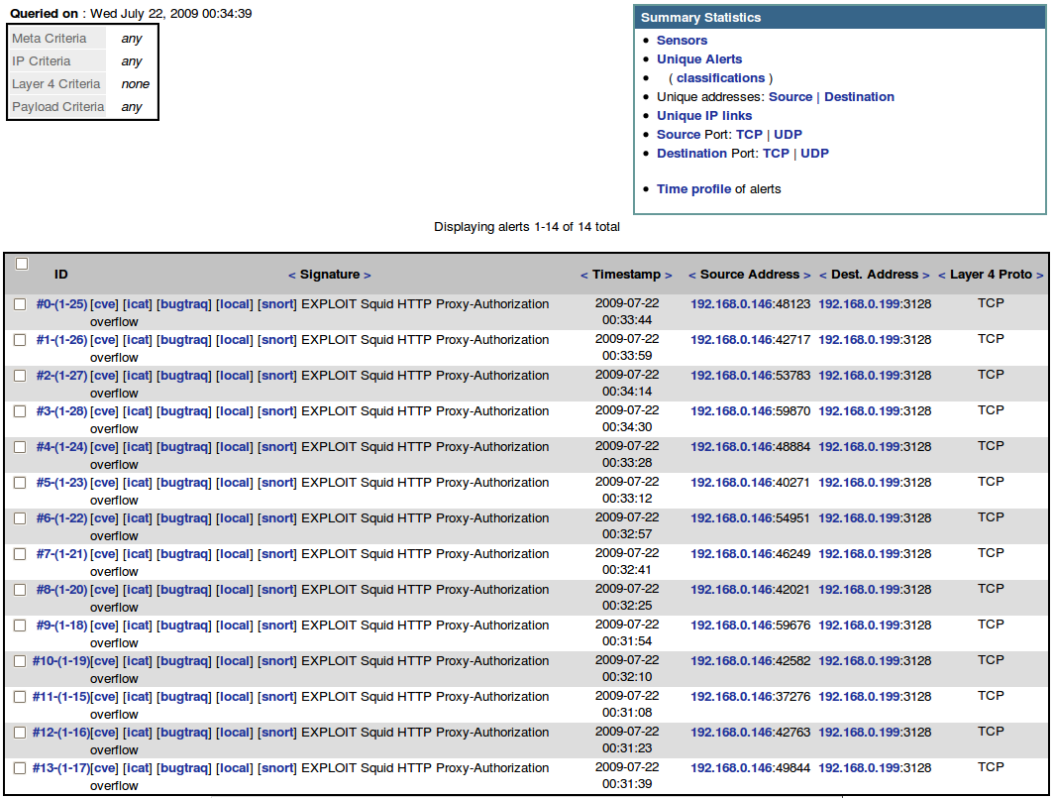


Figura 7.14: Consola de resultados en MSF en la PoC #1.



Queried on : Wed July 22, 2009 00:34:39

Meta Criteria	any
IP Criteria	any
Layer 4 Criteria	none
Payload Criteria	any

Summary Statistics

- Sensors
- Unique Alerts
- (classifications)
- Unique addresses: Source | Destination
- Unique IP links
- Source Port: TCP | UDP
- Destination Port: TCP | UDP
- Time profile of alerts

Displaying alerts 1-14 of 14 total

ID	< Signature >	< Timestamp >	< Source Address >	< Dest. Address >	< Layer 4 Proto >
#0-(1-25)	[cve] [icat] [bugtraq] [local] [snort] EXPLOIT Squid HTTP Proxy-Authorization overflow	2009-07-22 00:33:44	192.168.0.146:48123	192.168.0.199:3128	TCP
#1-(1-26)	[cve] [icat] [bugtraq] [local] [snort] EXPLOIT Squid HTTP Proxy-Authorization overflow	2009-07-22 00:33:59	192.168.0.146:42717	192.168.0.199:3128	TCP
#2-(1-27)	[cve] [icat] [bugtraq] [local] [snort] EXPLOIT Squid HTTP Proxy-Authorization overflow	2009-07-22 00:34:14	192.168.0.146:53783	192.168.0.199:3128	TCP
#3-(1-28)	[cve] [icat] [bugtraq] [local] [snort] EXPLOIT Squid HTTP Proxy-Authorization overflow	2009-07-22 00:34:30	192.168.0.146:59870	192.168.0.199:3128	TCP
#4-(1-24)	[cve] [icat] [bugtraq] [local] [snort] EXPLOIT Squid HTTP Proxy-Authorization overflow	2009-07-22 00:33:28	192.168.0.146:48884	192.168.0.199:3128	TCP
#5-(1-23)	[cve] [icat] [bugtraq] [local] [snort] EXPLOIT Squid HTTP Proxy-Authorization overflow	2009-07-22 00:33:12	192.168.0.146:40271	192.168.0.199:3128	TCP
#6-(1-22)	[cve] [icat] [bugtraq] [local] [snort] EXPLOIT Squid HTTP Proxy-Authorization overflow	2009-07-22 00:32:57	192.168.0.146:54951	192.168.0.199:3128	TCP
#7-(1-21)	[cve] [icat] [bugtraq] [local] [snort] EXPLOIT Squid HTTP Proxy-Authorization overflow	2009-07-22 00:32:41	192.168.0.146:46249	192.168.0.199:3128	TCP
#8-(1-20)	[cve] [icat] [bugtraq] [local] [snort] EXPLOIT Squid HTTP Proxy-Authorization overflow	2009-07-22 00:32:25	192.168.0.146:42021	192.168.0.199:3128	TCP
#9-(1-18)	[cve] [icat] [bugtraq] [local] [snort] EXPLOIT Squid HTTP Proxy-Authorization overflow	2009-07-22 00:31:54	192.168.0.146:59676	192.168.0.199:3128	TCP
#10-(1-19)	[cve] [icat] [bugtraq] [local] [snort] EXPLOIT Squid HTTP Proxy-Authorization overflow	2009-07-22 00:32:10	192.168.0.146:42582	192.168.0.199:3128	TCP
#11-(1-15)	[cve] [icat] [bugtraq] [local] [snort] EXPLOIT Squid HTTP Proxy-Authorization overflow	2009-07-22 00:31:08	192.168.0.146:37276	192.168.0.199:3128	TCP
#12-(1-16)	[cve] [icat] [bugtraq] [local] [snort] EXPLOIT Squid HTTP Proxy-Authorization overflow	2009-07-22 00:31:23	192.168.0.146:42763	192.168.0.199:3128	TCP
#13-(1-17)	[cve] [icat] [bugtraq] [local] [snort] EXPLOIT Squid HTTP Proxy-Authorization overflow	2009-07-22 00:31:39	192.168.0.146:49844	192.168.0.199:3128	TCP

Figura 7.15: Alertas generadas en BASE en la PoC #5.

Time	192.168.0.146 192.168.0.19	Comment
0,002	37276 > ndl-aas [SY (37276) (3128)	TCP: 37276 > ndl-aas [SYN] Seq=0 Win=5840 Len=0 MSS=1460
0,002	ndl-aas > 37276 [SY (37276) (3128)	TCP: ndl-aas > 37276 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
0,003	37276 > ndl-aas [AC (37276) (3128)	TCP: 37276 > ndl-aas [ACK] Seq=1 Ack=1 Win=5840 Len=0
0,007	GET http://ibxFRMWr (37276) (3128)	HTTP: GET http://ibxFRMWrDLroFoWSwCEvpEBlebWJuUFttOmcUBZThxapLiooFFVhrjoGjOMIbDeOXRlnvQzZhKfHcmvfji
0,007	ndl-aas > 37276 [AC (37276) (3128)	TCP: ndl-aas > 37276 [ACK] Seq=1 Ack=216 Win=6432 Len=0
0,008	[TCP segment of a r (37276) (3128)	TCP: [TCP segment of a reassembled PDU]
0,008	HTTP/1.0 403 Forbid (37276) (3128)	HTTP: HTTP/1.0 403 Forbidden (text/html)
0,008	GET http://ibxFRMWr (37276) (3128)	HTTP: GET http://ibxFRMWrDLroFoWSwCEvpEBlebWJuUFttOmcUBZThxapLiooFFVhrjoGjOMIbDeOXRlnvQzZhKfHcmvfji
0,008	ndl-aas > 37276 [AC (37276) (3128)	TCP: ndl-aas > 37276 [ACK] Seq=1541 Ack=864 Win=7764 Len=0
0,009	[TCP segment of a r (37276) (3128)	TCP: [TCP segment of a reassembled PDU]
0,009	HTTP/1.0 403 Forbid (37276) (3128)	HTTP: HTTP/1.0 403 Forbidden (text/html)
0,011	37276 > ndl-aas [RS (37276) (3128)	TCP: 37276 > ndl-aas [RST] Seq=216 Win=0 Len=0
0,011	37276 > ndl-aas [RS (37276) (3128)	TCP: 37276 > ndl-aas [RST] Seq=216 Win=0 Len=0
0,012	37276 > ndl-aas [RS (37276) (3128)	TCP: 37276 > ndl-aas [RST] Seq=864 Win=0 Len=0
0,012	37276 > ndl-aas [RS (37276) (3128)	TCP: 37276 > ndl-aas [RST] Seq=864 Win=0 Len=0
0,012	37276 > ndl-aas [RS (37276) (3128)	TCP: 37276 > ndl-aas [RST] Seq=864 Win=0 Len=0

Figura 7.16: Flujo de tráfico observado mediante Wireshark en la PoC #5.

7.4. Conclusiones

El objetivo de este capítulo ha sido verificar la respuesta de Snort ante ataques realizados con exploits específicos. Aunque finalmente se ha realizado un número de pruebas limitado, la unanimidad de los resultados obtenidos permiten asegurar con discreción que la respuesta del IDS es satisfactoria en cuanto a vulnerabilidades conocidas.

Cada día aparecen un gran número de nuevas vulnerabilidades y sus exploits correspondientes en sitios como *www.milw0rm.com*, sin embargo, conseguir un IDS capaz de detectar todos los nuevos exploits que se publican parece una tarea de dimensiones desproporcionadas.

La técnica más adecuada a la hora de probar la configuración de un IDS en un entorno real parece, por tanto, basar los ataques de prueba en el conocimiento que se tiene de la configuración de la red. Una vez conocidos todos los servicios activos, sería necesario conocer las vulnerabilidades de cada uno de ellos, localizar los exploits correspondientes y verificar la respuesta del IDS para cada uno de estos exploits.

Los resultados obtenidos ponen de manifiesto que cuando el resultado es correcto, la identificación es muy precisa a pesar de que no se haya conseguido explotar ninguno de los servicios instalados. Para conocer la respuesta del IDS no ha sido necesario que el exploit tuviera que provocar un fallo en el servicio atacado, a pesar de que en algunos casos se torna imprescindible la presencia del servicio para poder verificar la respuesta del IDS al ataque. Esto permite mostrar la dificultad a la hora de explotar un servicio con un exploit diseñado para una versión específica del servidor, en un sistema operativo concreto y en un entorno específico. No obstante, el trabajo realizado con el Framework Metasploit ha permitido comprobar su simplicidad de uso, lo que pone al alcance de cualquier individuo una herramienta de ataque preparada para actuar de forma maliciosa con escasos conocimientos del proceso que se está llevando a cabo.

El hecho de que Snort presente una respuesta satisfactoria ante los módulos incluidos por defecto en Metasploit, la mayoría de ellos exploits de vulnerabilidades muy conocidas, representa una característica muy positva y dado que el éxito de este entorno de exploiting continúa creciendo gracias a su sencillez de uso, una tarea en la que el equipo de desarrollo del IDS debería centrar parte de sus esfuerzos de forma continuada.

Capítulo 8

Conclusiones

Conclusiones generales

A lo largo del desarrollo de este proyecto se ha tratado de comprobar la respuesta del IDS Snort a diversos ataques, la efectividad de sus preprocesadores y su motor de reglas y se ha intentado verificar la rapidez de los procesos de actualización involucrados en su desarrollo. Sin embargo, a pesar de los posibles resultados e independientemente del sentido positivo o negativo de los mismos, una conclusión subyace a todo el trabajo realizado: en cualquier red desplegada en la actualidad, el IDS es una herramienta imprescindible.

Snort ha demostrado ser un sistema relativamente sencillo de usar una vez instalado y configurado, y su reconocido estatus como IDS de referencia se encuentra justificado. El IDS del equipo Sourcefire permite a usuarios inexpertos poder garantizar un mínimo de seguridad ante ataques conocidos y a usuarios avanzados establecer un sistema que, en máquinas con potencia suficiente, puede asegurar entornos de red de un tamaño considerable.

A la hora de definir y configurar el entorno para realizar las diferentes pruebas de concepto, cabe destacar, la versatilidad de los sistemas virtuales utilizados. El uso de máquinas virtuales permite simular entornos con unas características especificas, en este caso, entornos donde se encuentran accesibles servidores vulnerables. Todo esto a un coste reducido que evita la necesidad de utilizar una infraestructura vulnerable real, específicamente dedicada a pruebas de verificación de ataques. Las máquinas virtuales permiten trabajar, por tanto, en entornos de riesgo sin necesidad de comprometer la seguridad de una red real.

Retardos de Actualización

En el estudio llevado a cabo sobre retardos de actualización se ha verificado la existencia de los mismos, tanto en la actualización de las reglas de Snort frente a Nessus como respecto al momento de publicación de las vulnerabilidades, asociados éstos a la aparición de sus identificadores CVE y Bugtraq.

Los resultados y medidas obtenidos para los diferentes retardos indican que en algunos casos y para algunas vulnerabilidades pueden ser considerablemente altos. Posteriormente se han comparado dichos retardos con las ventanas de vulnerabilidad de las aplicaciones, obteniendo resultados que permiten asegurar que una política adecuada de actualización de las aplicaciones sigue siendo el método más efectivo para evitar ataques a vulnerabilidades en los sistemas. No obstante, se comprueba que en algunos casos, el IDS es capaz de detectar el ataque con cierta antelación a la publicación de un parche para el software vulnerable, por lo que su utilización y adecuada actualización si bien no constituye una garantía absoluta, si supone un elemento indispensable en cualquier plan de seguridad.

Respuesta ante ataques de reconocimiento

Respecto a los ataques de reconocimiento y el papel del preprocesador sfportscan se han llevado a cabo pruebas de concepto con ataques convencionales, lo que ha proporcionado unos resultados considerablemente buenos en cuanto a la respuesta del IDS. A partir de estas pruebas se abre todo un campo de experimentación sobre técnicas de evasión más complejas para *escáners* de puertos. El uso de técnicas que imposibilitan la detección del atacante como el *idle scan* ponen de manifiesto algunas limitaciones de los sistemas IDS pasivos y la necesidad de sistemas activos o IPS que permitan rastrear y obtener mayor información de los atacantes y sus acciones.

Es importante reseñar que los mecanismos implementados en Snort para la detección de ataques de reconocimiento buscan específicamente identificar aquellos análisis que pueden ser realizados por la herramienta que se ha utilizado para los mismos en las pruebas de concepto: Nmap, un estándar de facto entre este tipo de herramientas. Por esta razón, los resultados son satisfactorios incluso cuando se realizan dichos análisis combinándolos con técnicas para no ser detectados. Posibles ramificaciones de este estudio tendrían como objetivo analizar, no sólo la respuesta ante otras herramientas menos conocidas, sino también ante una serie de

técnicas más complejas y específicas.

Respuesta frente a escáner de vulnerabilidades

En el caso del VDS Nessus, se ha comprobado por qué es un estándar en sistemas de auditoria. Sus periodos de actualización son rápidos, existen versiones para todos los sistemas y es muy sencillo de usar. Incluso para redes sencillas con usuarios inexpertos puede ser una herramienta muy eficaz a la hora de conocer el estado de sus sistemas. A la hora de realizar las pruebas para llevar a cabo el proceso de evaluación de Snort se han generado muchos falsos positivos debido a los escáners de puertos que el escáner Nessus realiza previos a cada comprobación de vulnerabilidades, de ahí la necesidad de probar la configuración tan sólo con ataques por medio de *exploits*, los cuales suponen la verdadera amenaza a la integridad de los servicios.

Respuesta ante exploits

El *Framework Metasploit* destaca, sobre todo, por su facilidad de uso y aunque los módulos incluidos por defecto para sistemas linux son escasos, para otros sistemas como Microsoft Windows existe un gran número de *exploits* y se continúan añadiendo en cada nueva versión que es publicada.

Desde el punto de vista de la actualización cabe destacar que aunque la publicación de nuevas versiones de Metasploit se hace tras un periodo de tiempo considerable, solamente atacantes sin conocimiento se limitarán a utilizar los *exploits* incluidos en cada versión. Metasploit permite añadir y utilizar cualquier *exploit* que haya sido publicado o creado por un usuario. De este modo, conocer el retardo de actualización de Snort ante la herramienta Metasploit lleva de vuelta a la necesidad de analizar la rapidez de actualización del motor de reglas del IDS frente a nuevas vulnerabilidades publicadas susceptibles de ser explotadas. Esta comprobación es la que se intentado llevar a cabo con el análisis del retardo de actualización frente a los indicadores CVE y Bugtraq.

Dificultades

A lo largo, tanto del estudio teórico de cada una de las herramientas y sus características de funcionamiento, como del desarrollo de las pruebas de concepto, se han encontrado una

serie de dificultades que pueden ser destacadas sobre las demás.

Comenzando por la instalación y definición del entorno de pruebas y a pesar de que una vez instalado, su funcionamiento resulta relativamente sencillo, las mayores dificultades encontradas atañen a la instalación y configuración del IDS y el demás software necesario: mysql, php, apache, base, nessus, nmap. o metasploit. Partiendo de un desconocimiento considerable en el manejo y funcionamiento de todas las herramientas, el proceso de aprendizaje de todas ellas ha resultado la parte más compleja.

La configuración de un sistema vulnerable para realizar las distintas pruebas también ha resultado complicado. La búsqueda de servicios vulnerables, antiguos o desactualizados que poder probar ha resultado complicado ya que todos los desarrolladores tratan de conseguir sistemas lo más seguros posible. De este modo, intentar configurar un sistema en esa franja de inseguridad en la que los sistemas tienden a no encontrarse no ha resultado sencillo. Durante el desarrollo del proyecto se han conocido otras soluciones ideadas con este fin como el sistema *DamnVulnerable Linux (http://www.damnvulnerablelinux.org/)* que pueden facilitar esta tarea para otros estudios que traten de probar la configuración de un IDS y su respuesta ante ataques.

Respecto al desarrollo de las pruebas, cabe destacar que el uso de *exploits*, que indudablemente es la mejor solución para probar el IDS, no siempre permite encontrar servicios realmente vulnerables y aunque sólo interesa conocer la respuesta del IDS ante los paquetes de código malicioso, sería interesante conocer si ante un ataque exitoso, el IDS puede aportar algo más de información.

Como se ha comentado previamente, ha resultado difícil conseguir información para correlar entre diferentes identificadores, plugins, vulnerabilidades y reglas. Aunque existen algunos intentos de centralizar toda esta información como la base de datos *The Open Source Vulnerability Database http/www.osdbv.com*, no siempre se encuentra completa y no permite obtener un perfil adecuado y con suficiente información de varios servicios de forma automática.

Trabajo Futuro

Desde el punto de vista de la gestión e identificación de vulnerabilidades, el trabajo continuo con las mismas durante el desarrollo del estudio y las pruebas ha permitido comprobar que aunque es posible encontrar información sobre una vulnerabilidad concreta, sobre todo si

ésta se encuentra correctamente definida por un identificador tipo CVE, Bugtraq, la información relativa a herramientas que puedan detectarla, probarla o explotarla es muy limitado y no existe una base de datos de calidad donde cada una de las vulnerabilidades se encuentre asociada a un contexto e interrelacionada con las diferentes herramientas.

En el caso del escáner de vulnerabilidades Nessus, los periodos de actualización son rápidos y se publican nuevos plugins cada día, sin embargo, el número de vulnerabilidades existentes sigue siendo mayor. Nessus ha demostrado su eficacia para detectar vulnerabilidades conocidas, pero a pesar de esto, las pruebas realizadas muestran que cuando se busca verificar la configuración del IDS las herramientas de explotación directa representan una mejor opción. Solo ante verdaderas condiciones de ataque podremos conocer con exactitud la respuesta real del sistema de detección de intrusión.

Ocurre algo similar con el motor de reglas de Snort, que aunque amplio y actualizado de forma algo más lenta, no alcanza a cubrir todas las vulnerabilidades que aparecen continuamente. En el caso de Snort, existe la opción para el usuario de crear reglas específicas para una vulnerabilidad, sin embargo, el análisis de las ventanas de vulnerabilidades de las aplicaciones en comparación con las del escáner y el IDS, demuestran que una actualización adecuada de los servicios a proteger es siempre la opción mas rápida y segura. La actualización sistemática supone, además, un esfuerzo mucho menor que la creación de reglas individuales para cada servicio vulnerable cuyos ataques todavía no pueden ser detectados por Snort. En cualquier caso y al margen de las reglas oficiales publicadas por el equipo de desarrollo de Snort, existen páginas como *http://www.bleedingsnort.com/* donde se pueden conseguir reglas creadas por administradores independientes, aunque hay que tener en cuenta que cada entorno es muy específico y las reglas pueden estar diseñadas para una situación concreta, de modo que ante la ineficacia de un desarrollo individualizado y especializado de reglas, la mejor opción es la contextualización de las mismas.

A partir de estas ideas, se puede plantear la necesidad de diseñar una herramienta de auditoria que pudiera correlar de forma automática las bases de datos de Snort, Nessus, Metasploit, actualizaciones de software de distintos desarrolladores, identificadores de vulnerabilidades y el estado de los servicios activos en la red. De este modo, el usuario podría conocer que vulnerabilidades requieren atención especial por no existir actualizaciones de software o herramientas de detección de ataques.

Una herramienta de este tipo requeriría que los diversos desarrolladores estandarizaran

sus alertas, plugins o *exploits* en un archivo accesible y analizable para cada una de ellas, por lo que parece una mejor idea la creación de una base de datos única en Internet y de un pequeño software cliente capaz de acceder a la misma y generar información del contexto. La base de datos permitiría correlar no sólo vulnerabilidades, si no versiones de software, sistemas operativos, presencia de esa vulnerabilidad en un escaner o si es posible la detección con una regla de un IDS específico.

Si el equipo de desarrollo de un IDS publica una regla para una nueva vulnerabilidad en un entorno determinado, sólo necesitaría añadir esos datos a la base de datos *online* y en un análisis de contexto realizado por el software cliente, la información de que dicho IDS detecta ese ataque sería tenida en cuenta para informar al usuario de la acción más conveniente. El software cliente debería conocer qué servicios existen en la red, ya sea por medio de un escáner de red o desde el propio sistema local. También debería ser posible obtener información relativa a vulnerabilidades, parches y herramientas al solicitar información especifica sobre un servicio.

La idea de una base de datos universal de vulnerabilidades con información de contexto y herramientas permitiría desarrollar al máximo la idea del *full disclosure* tan alabada por algunos y criticada por otros. Si la información se encuentra bien estructurada y es accesible de forma universal, beneficia a todos los usuarios. En caso contrario, no sólo no beneficia sino que puede ayudar a los atacantes a utilizar técnicas que no son ampliamente conocidas.

Opiniones Personales

Desde el punto de vista personal, puedo asegurar que he acertado a la hora de elegir este proyecto de fin de carrera. Durante todo el desarrollo, he mantenido la sensación de querer seguir profundizando en cada concepto, en cada técnica, en el uso de cada herramienta. Este estudio sienta las bases para estudios más específicos que incluyan el análisis de técnicas más complejas de evasión o explotación y aunque la extensión de este trabajo es limitada, me ha ayudado a afianzar la fuerte vocación por el mundo de la seguridad que me gustaría desarrollar en el mundo laboral.

Este proyecto me ha permitido, siempre con esfuerzo y motivación, pasar del desconocimiento a una situación donde he tenido que trabajar con herramientas, conceptos y soluciones que se están utilizando de forma muy práctica en entornos reales. El mundo de la seguridad es

uno de los más dinámicos y cambiantes que existen y es precisamente esa necesidad de aprendizaje constante uno de los mayores atractivos tanto del trabajo realizado como del trabajo por realizar. Este proyecto me ha permitido poner de manifiesto y analizar características y el funcionamiento de diversas herramientas, pero por encima de todo siento que me ha permitido aprender con una gran motivación y eso es, a todas luces, un éxito.

APÉNDICES

APÉNDICE A

Tablas de vulnerabilidades para cálculo del retardo de actualización

Disclosure	Bugtraq	Bugtraq Last Revision Date	CVE	CVE Last Revision Date	Snort Rule	Snort Date	Nessus Plugin	Nessus Date	Snort-Bugtraq Delay	Snort-CVE Delay	Snort-Nessus Delay
					15423 <-> SPECIFIC-THREATS Clampi virus communication detected (specific-threats.rules, High)	27-Mar-2009					
5-Nov-2008	32123	7-Nov-2008	2008-6301	13-Mar-2009	15424 <-> WEB-PHP phpBB mod shoutbox sql injection attempt (web-php.rules, High)	27-Mar-2009	No		140	14	
8-Dec-2008	32701**	11-Dec-2008	2008-6314	27-Feb-2009	15425 <-> WEB-PHP phpBB mod tag board sql injection attempt (web-php.rules, High)	27-Mar-2009	No		0	30	
5-Mar-2009	33990*	4-Mar-2009	2009-0771*	5-Mar-2009	15428 <-> WEB-CLIENT Mozilla Firefox SVG data processing memory corruption attempt (web-client.rules, High)	27-Mar-2009	Firefox < 3.0.7 Multiple Vulnerabilities 35778	5-Mar-2009	23	22	22
					15429 <-> CONTENT-REPLACE Yahoo Messenger deny outbound login attempt (content-replace.rules, High)	27-Mar-2009					
23-Mar-2009	34250*	24-Mar-2009	2009-1217	1-Apr-2009	15430 <-> WEB-CLIENT Microsoft EMF+ GpFont. SetData buffer overflow attempt (web-client.rules, High)	27-Mar-2009	No		3	-4	
25-Mar-2009	34235*	25-Mar-2009	2009-1169*	27-Mar-2009	15431 <-> SPECIFIC-THREATS Firefox 3 xsl parsing heap overflow attempt (specific-threats.rules, High)	27-Mar-2009	Firefox < 3.0.8 Multiple Vulnerabilities 36045	30-Mar-2009	2	0	-3
2-Apr-2008	28845**	18-Apr-2008	2008-4769	18-Mar-2009	15432 <-> WEB-PHP wordpress cat parameter arbitrary file execution attempt (web-php.rules, High)	27-Mar-2009	WordPress index.php cat Parameter Local File Inclusion 32080**	29-Apr-2008	0	9	0
19-May-2004	10386**	17-Oct-2007	2004-0397	5-Sep-2008	15388 <-> EXPLOIT Subversion 1.0.2 get-dated-rev buffer overflow over http attempt (exploit.rules, High)	17-Mar-2009	Subversion remote Buffer Overflow 12261**	7-Jul-2004	0	192	0
					15415 <-> CONTENT-REPLACE AIM or ICQ deny unencrypted login connection (content-replace.rules, High)	17-Mar-2009					
					15416 <-> CONTENT-REPLACE ICQ deny http proxy login (content-replace.rules, High)	17-Mar-2009					
					15417 <-> CONTENT-REPLACE AIM deny server certificate for encrypted login (content-replace.rules, High)	17-Mar-2009					
					15418 <-> CHAT AIM server certificate for encrypted login (chat.rules, High)	17-Mar-2009					
					15420 <-> CONTENT-REPLACE MSN deny login (content-replace.rules, High)	17-Mar-2009					
					15421 <-> CONTENT-REPLACE AIM or ICQ deny login for unencrypted connection (content-replace. rules, High)	17-Mar-2009					
25-May-2007	24165**	15-Nov-2007	2007-2881	5-Sep-2008	15422 <-> SPECIFIC-THREATS Sun One web proxy server overflow attempt (specific-threats.rules, High)	17-Mar-2009	No		0	192	
13-Jun-2006	18381**	28-Jun-2006	2006-1193	5-Sep-2008	15367 <-> SMTP outlook web access script injection attempt (smtp.rules, High)	3-Mar-2009	MS06-029: Vulnerability in Microsoft Exchange Server Running Outlook Web Access Could Allow Script Injection (912442) 21695**	16-Jun-2006	0	178	0
	33842	20-Feb-2009			15368 <-> WEB-ACTIVEX FathFTP ActiveX clsid access (web-activex.rules, High)	3-Mar-2009	No		13		
	33842	20-Feb-2009			15369 <-> WEB-ACTIVEX FathFTP ActiveX clsid unicode access (web-activex.rules, High)	3-Mar-2009	No		13		
	33842	20-Feb-2009			15370 <-> WEB-ACTIVEX FathFTP ActiveX function call access (web-activex.rules, High)	3-Mar-2009	No		13		
	33842	20-Feb-2009			15371 <-> WEB-ACTIVEX FathFTP ActiveX function call unicode access (web-activex. rules, High)	3-Mar-2009	No		13		
	33867	5-Mar-2009			15372 <-> WEB-ACTIVEX iDefense COMRaider ActiveX clsid access (web-activex. rules, High)	3-Mar-2009	No		-2		
	33867	5-Mar-2009			15373 <-> WEB-ACTIVEX iDefense COMRaider ActiveX clsid unicode access (web-activex.rules, High)	3-Mar-2009	No		-2		
	33867	5-Mar-2009			15374 <-> WEB-ACTIVEX iDefense COMRaider ActiveX function call access (web-activex.rules, High)	3-Mar-2009	No		-2		

	33867	5-Mar-2009			15375 <-> WEB-ACTIVEX iDefense COMRaider ActiveX function call unicode access (web-activex.rules, High)	3-Mar-2009	No		-2		
25-Feb-2009	33920	26-Feb-2009	2009-0811	5-Mar-2009	15376 <-> WEB-ACTIVEX Sopcast SopCore ActiveX clsid access (web-activex.rules, High)	3-Mar-2009	No		7	-2	
25-Feb-2009	33920	26-Feb-2009	2009-0811	5-Mar-2009	15377 <-> WEB-ACTIVEX Sopcast SopCore ActiveX clsid unicode access (web-activex. rules, High)	3-Mar-2009	No		7	-2	
25-Feb-2009	33920	26-Feb-2009	2009-0811	5-Mar-2009	15378 <-> WEB-ACTIVEX Sopcast SopCore ActiveX function call access (web-activex.rules, High)	3-Mar-2009	No		7	-2	
25-Feb-2009	33920	26-Feb-2009	2009-0811	5-Mar-2009	15379 <-> WEB-ACTIVEX Sopcast SopCore ActiveX function call unicode access (web-activex.rules, High)	3-Mar-2009	No		7	-2	
24-Feb-2009	33918	27-Feb-2009	2009-0208	27-Feb-2009	15380 <-> WEB-ACTIVEX HP Virtual Rooms v7 ActiveX clsid access (web-activex.rules, High)	3-Mar-2009	HP Virtual Rooms Client < 7.0.1 ActiveX Control Dangerous Methods 35804	9-Mar-2009	6	6	-6
24-Feb-2009	33918	27-Feb-2009	2009-0208	27-Feb-2009	15381 <-> WEB-ACTIVEX HP Virtual Rooms v7 ActiveX clsid unicode access (web-activex. rules, High)	3-Mar-2009	HP Virtual Rooms Client < 7.0.1 ActiveX Control Dangerous Methods 35804	9-Mar-2009	6	6	-6
2-Oct-2007	25898**	19-Mar-2008	2007-4568	5-Sep-2008	15382 <-> SPECIFIC-THREATS X.Org X Font Server QueryXBitmaps and QueryXExtents Handlers integer overflow attempt (specific-threats.rules, High)	3-Mar-2009	Mac OS X Security Update 2008-001 30254**	12-Feb-2008	0	178	0
18-Oct-2007	26132**	24-Apr-2008	2007-5339	5-Sep-2008	15383 <-> SPECIFIC-THREATS Mozilla Firefox XBL Event Handler Tags Removal memory corruption attempt (specific-threats.rules, High)	3-Mar-2009	Firefox < 2.0.0.8 27521**	19-Oct-2007	0	178	0
5-Nov-2007	26345**	15-Nov-2007	2007-4676	15-Nov-2008	15384 <-> WEB-CLIENT Apple QuickTime pict image poly structure memory corruption attempt (web-client.rules, High)	3-Mar-2009	QuickTime < 7.3 (Mac OS X) 27625**	6-Nov-2007	0	108	0
13-Jan-2009	33299*	15-Jan-2009	2009-0241*	21-Jan-2009	15364 <-> EXPLOIT Ganglia Meta Daemon process_path stack buffer overflow attempt (exploit.rules, High)	27-Feb-2009	FreeBSD : ganglia – buffer overflow vulnerability (2203) 35564	1-Feb-2009	42	36	26
19-Feb-2009	33751*	19-Feb-2009	2009-0658*	20-Feb-2009	15358 <-> SMTP Adobe PDF JBIG2 remote code execution attempt (smtp.rules, High)	24-Feb-2009	Adobe Reader < 9.1 / 8.1.4 / 7.1.1 35821	11-Mar-2009	5	4	-17
19-Feb-2009	33751*	19-Feb-2009	2009-0658*	20-Feb-2009	15359 <-> SMTP suspicious pdf file sent via email (smtp. rules, High)	24-Feb-2009	Adobe Reader < 9.1 / 8.1.4 / 7.1.1 35821	11-Mar-2009	5	4	-17
19-Feb-2009	33751*	19-Feb-2009	2009-0658*	20-Feb-2009	15360 <-> SMTP suspicious pdf file sent via email (smtp. rules, High)	24-Feb-2009	Adobe Reader < 9.1 / 8.1.4 / 7.1.1 35821	11-Mar-2009	5	4	-17
19-Feb-2009	33751*	19-Feb-2009	2009-0658*	20-Feb-2009	15361 <-> POLICY pdf file sent via email (policy.rules, High)	24-Feb-2009	Adobe Reader < 9.1 / 8.1.4 / 7.1.1 35821	11-Mar-2009	5	4	-17
					15307 <-> WEB-ACTIVEX Microsoft Animation Control ActiveX clsid access (web-activex.rules, High)	20-Feb-2009					
					15308 <-> WEB-ACTIVEX Microsoft Animation Control ActiveX clsid unicode access (web-activex.rules, High)	20-Feb-2009					
					15309 <-> WEB-ACTIVEX Microsoft Animation Control ActiveX function call access (web-activex.rules, High)	20-Feb-2009					
					15310 <-> WEB-ACTIVEX Microsoft Animation Control ActiveX function call unicode access (web-activex.rules, High)	20-Feb-2009					
					15311 <-> WEB-ACTIVEX Research In Motion AxLoader ActiveX clsid access (web-activex.rules, High)	20-Feb-2009					
					15312 <-> WEB-ACTIVEX Research In Motion AxLoader ActiveX clsid unicode access (web-activex.rules, High)	20-Feb-2009					
					15313 <-> WEB-ACTIVEX Research In Motion AxLoader ActiveX function call access (web-activex.rules, High)	20-Feb-2009					
					15314 <-> WEB-ACTIVEX Research In Motion AxLoader ActiveX function call unicode access (web-activex.rules, High)	20-Feb-2009					

					15315 <-> WEB-ACTIVEX Akamai DownloadManager ActiveX clsid access (web-activex.rules, High)	20-Feb-2009					
					15316 <-> WEB-ACTIVEX Akamai DownloadManager ActiveX clsid unicode access (web-activex.rules, High)	20-Feb-2009					
					15317 <-> WEB-ACTIVEX Akamai DownloadManager ActiveX function call access (web-activex.rules, High)	20-Feb-2009					
					15318 <-> WEB-ACTIVEX Akamai DownloadManager ActiveX function call unicode access (web-activex.rules, High)	20-Feb-2009					
	33726	10-Feb-2009			15330 <-> WEB-ACTIVEX Nokia Phoenix Service 1 ActiveX clsid access (web-activex.rules, High)	20-Feb-2009	No		**10**		
	33726	10-Feb-2009			15331 <-> WEB-ACTIVEX Nokia Phoenix Service 1 ActiveX clsid unicode access (web-activex.rules, High)	20-Feb-2009	No		**10**		
	33726	10-Feb-2009			15332 <-> WEB-ACTIVEX Nokia Phoenix Service 2 ActiveX clsid access (web-activex.rules, High)	20-Feb-2009	No		**10**		
	33726	10-Feb-2009			15333 <-> WEB-ACTIVEX Nokia Phoenix Service 2 ActiveX clsid unicode access (web-activex.rules, High)	20-Feb-2009	No		**10**		
16-Feb-2009	33782	17-Feb-2009	2009-0865	10-Mar-2009	15334 <-> WEB-ACTIVEX GeoVision LiveX 7000 ActiveX clsid access (web-activex.rules, High)	20-Feb-2009	No		**3**	**-20**	
16-Feb-2009	33782	17-Feb-2009	2009-0865	10-Mar-2009	15335 <-> WEB-ACTIVEX GeoVision LiveX 7000 ActiveX clsid unicode access (web-activex.rules, High)	20-Feb-2009	No		**3**	**-20**	
16-Feb-2009	33782	17-Feb-2009	2009-0865	10-Mar-2009	15336 <-> WEB-ACTIVEX GeoVision LiveX 7000 ActiveX function call access (web-activex.rules, High)	20-Feb-2009	No		**3**	**-20**	
16-Feb-2009	33782	17-Feb-2009	2009-0865	10-Mar-2009	15337 <-> WEB-ACTIVEX GeoVision LiveX 7000 ActiveX function call unicode access (web-activex.rules, High)	20-Feb-2009	No		**3**	**-20**	
16-Feb-2009	33782	17-Feb-2009	2009-0865	10-Mar-2009	15338 <-> WEB-ACTIVEX GeoVision LiveX 8120 ActiveX clsid access (web-activex.rules, High)	20-Feb-2009	No		**3**	**-20**	
16-Feb-2009	33782	17-Feb-2009	2009-0865	10-Mar-2009	15339 <-> WEB-ACTIVEX GeoVision LiveX 8120 ActiveX clsid unicode access (web-activex.rules, High)	20-Feb-2009	No		**3**	**-20**	
16-Feb-2009	33782	17-Feb-2009	2009-0865	10-Mar-2009	15340 <-> WEB-ACTIVEX GeoVision LiveX 8120 ActiveX function call access (web-activex.rules, High)	20-Feb-2009	No		**3**	**-20**	
16-Feb-2009	33782	17-Feb-2009	2009-0865	10-Mar-2009	15341 <-> WEB-ACTIVEX GeoVision LiveX 8120 ActiveX function call unicode access (web-activex.rules, High)	20-Feb-2009	No		**3**	**-20**	
16-Feb-2009	33782	17-Feb-2009	2009-0865	10-Mar-2009	15342 <-> WEB-ACTIVEX GeoVision LiveX 8200 ActiveX clsid access (web-activex.rules, High)	20-Feb-2009	No		**3**	**-20**	
16-Feb-2009	33782	17-Feb-2009	2009-0865	10-Mar-2009	15343 <-> WEB-ACTIVEX GeoVision LiveX 8200 ActiveX clsid unicode access (web-activex.rules, High)	20-Feb-2009	No		**3**	**-20**	
16-Feb-2009	33782	17-Feb-2009	2009-0865	10-Mar-2009	15344 <-> WEB-ACTIVEX GeoVision LiveX 8200 ActiveX function call access (web-activex.rules, High)	20-Feb-2009	No		**3**	**-20**	
16-Feb-2009	33782	17-Feb-2009	2009-0865	10-Mar-2009	15345 <-> WEB-ACTIVEX GeoVision LiveX 8200 ActiveX function call unicode access (web-activex.rules, High)	20-Feb-2009	No		**3**	**-20**	
30-Jan-2009	33535	2-Feb-2009	2009-0465*	10-Feb-2009	15346 <-> WEB-ACTIVEX Synactis ALL In-The-Box ActiveX clsid access (web-activex.rules, High)	20-Feb-2009	No		**18**	**10**	
30-Jan-2009	33535	2-Feb-2009	2009-0465*	10-Feb-2009	15347 <-> WEB-ACTIVEX Synactis ALL In-The-Box ActiveX clsid unicode access (web-activex.rules, High)	20-Feb-2009	No		**18**	**10**	
30-Jan-2009	33535	2-Feb-2009	2009-0465*	10-Feb-2009	15348 <-> WEB-ACTIVEX Synactis ALL In-The-Box ActiveX function call access (web-activex.rules, High)	20-Feb-2009	No		**18**	**10**	
30-Jan-2009	33535	2-Feb-2009	2009-0465*	10-Feb-2009	15349 <-> WEB-ACTIVEX Synactis ALL In-The-Box ActiveX function call unicode access (web-activex.rules, High)	20-Feb-2009	No		**18**	**10**	

					15350 <-> WEB-ACTIVEX Web on Windows ActiveX clsid access (web-activex.rules, High)	20-Feb-2009					
					15351 <-> WEB-ACTIVEX Web on Windows ActiveX clsid unicode access (web-activex. rules, High)	20-Feb-2009					
					15352 <-> WEB-ACTIVEX Web on Windows ActiveX function call access (web-activex.rules, High)	20-Feb-2009					
					15353 <-> WEB-ACTIVEX Web on Windows ActiveX function call unicode access (web-activex.rules, High)	20-Feb-2009					
19-Feb-2009	33751*	19-Feb-2009	2009-0658*	20-Feb-2009	15356 <-> SMTP Adobe PDF JBIG2 remote code execution attempt (smtp.rules, High)	20-Feb-2009	Adobe Reader < 9.1 / 8.1.4 / 7.1.1 35821	11-Mar-2009	1	0	-21
19-Feb-2009	33751*	19-Feb-2009	2009-0658*	20-Feb-2009	15357 <-> WEB-CLIENT Adobe PDF JBIG2 remote code execution attempt (web-client.rules, High)	20-Feb-2009	Adobe Reader < 9.1 / 8.1.4 / 7.1.1 35821	11-Mar-2009	1	0	-21
	33233	17-Jan-2009			15228 <-> WEB-ACTIVEX Ciansoft PDFBuilderX ActiveX clsid access (web-activex. rules, High)	3-Feb-2009	No		16		
	33233	17-Jan-2009			15229 <-> WEB-ACTIVEX Ciansoft PDFBuilderX ActiveX clsid unicode access (web-activex.rules, High)	3-Feb-2009	No		16		
	23811;33238; 33243; 33245	27-Jan-2009	2007-2588**	15-Nov-2008	15230 <-> WEB-ACTIVEX Office Viewer 2 ActiveX clsid access (web-activex.rules, High)	3-Feb-2009	No		6	0	
	23811;33238; 33243; 33245	27-Jan-2009	2007-2588**	15-Nov-2008	15231 <-> WEB-ACTIVEX Office Viewer 2 ActiveX clsid unicode access (web-activex. rules, High)	3-Feb-2009	No		6	0	
14-Jan-2009	33272	16-Jan-2009	2009-0134	29-Jan-2009	15232 <-> WEB-ACTIVEX Easy Grid ActiveX clsid access (web-activex.rules, High)	3-Feb-2009	No		17	4	
14-Jan-2009	33272	16-Jan-2009	2009-0134	29-Jan-2009	15233 <-> WEB-ACTIVEX Easy Grid ActiveX clsid unicode access (web-activex. rules, High)	3-Feb-2009	No		17	4	
14-Jan-2009	33272	16-Jan-2009	2009-0134	29-Jan-2009	15234 <-> WEB-ACTIVEX Easy Grid ActiveX function call access (web-activex.rules, High)	3-Feb-2009	No		17	4	
14-Jan-2009	33272	16-Jan-2009	2009-0134	29-Jan-2009	15235 <-> WEB-ACTIVEX Easy Grid ActiveX function call unicode access (web-activex. rules, High)	3-Feb-2009	No		17	4	
23-Jan-2009	33408	2-Feb-2009	2008-5260	29-Jan-2009	15243 <-> WEB-ACTIVEX AXIS Camera ActiveX clsid access (web-activex.rules, High)	3-Feb-2009	AxisCamControl ActiveX Control Buffer Overflow Vulnerability 35454	24-Jan-2009	1	4	9
23-Jan-2009	33408	2-Feb-2009	2008-5260	29-Jan-2009	15244 <-> WEB-ACTIVEX AXIS Camera ActiveX clsid unicode access (web-activex. rules, High)	3-Feb-2009	AxisCamControl ActiveX Control Buffer Overflow Vulnerability 35454	24-Jan-2009	1	4	9
23-Jan-2009	33408	2-Feb-2009	2008-5260	29-Jan-2009	15245 <-> WEB-ACTIVEX AXIS Camera ActiveX function call access (web-activex.rules, High)	3-Feb-2009	AxisCamControl ActiveX Control Buffer Overflow Vulnerability 35454	24-Jan-2009	1	4	9
23-Jan-2009	33408	2-Feb-2009	2008-5260	29-Jan-2009	15246 <-> WEB-ACTIVEX AXIS Camera ActiveX function call unicode access (web-activex.rules, High)	3-Feb-2009	AxisCamControl ActiveX Control Buffer Overflow Vulnerability 35454	24-Jan-2009	1	4	9
	33345	20-Jan-2009			15247 <-> WEB-ACTIVEX JamDTA ActiveX clsid access (web-activex.rules, High)	3-Feb-2009	No		13		
	33345	20-Jan-2009			15248 <-> WEB-ACTIVEX JamDTA ActiveX clsid unicode access (web-activex.rules, High)	3-Feb-2009	No		13		
	33348, 33349	20-Jan-2009			15249 <-> WEB-ACTIVEX SmartVMD ActiveX clsid access (web-activex.rules, High)	3-Feb-2009	No		13		
	33348, 33349	20-Jan-2009			15250 <-> WEB-ACTIVEX SmartVMD ActiveX clsid unicode access (web-activex. rules, High)	3-Feb-2009	No		13		
	33318	19-Jan-2009			15251 <-> WEB-ACTIVEX MetaProducts MetaTreeX ActiveX clsid access (web-activex.rules, High)	3-Feb-2009	No		14		
	33318	19-Jan-2009			15252 <-> WEB-ACTIVEX MetaProducts MetaTreeX ActiveX clsid unicode access (web-activex.rules, High)	3-Feb-2009	No		14		
	33318	19-Jan-2009			15253 <-> WEB-ACTIVEX MetaProducts MetaTreeX ActiveX function call access (web-activex.rules, High)	3-Feb-2009	No		14		

	33318	19-Jan-2009			15254 <-> WEB-ACTIVEX MetaProducts MetaTreeX ActiveX function call unicode access (web-activex.rules, High)	3-Feb-2009	No		**14**		
24-Jan-2007	22196, 33469	28-Jan-2009	2007-0018	2-Jan-2009	15265 <-> WEB-ACTIVEX NCTAudioFile2 ActiveX function call unicode access (web-activex.rules, High)	3-Feb-2009	No		**5**	**31**	
26-Jan-2009	33451	28-Jan-2009	2008-4924*, 2009-0298	29-Jan-2009	15266 <-> WEB-ACTIVEX MW6 Technologies Barcode ActiveX clsid access (web-activex.rules, High)	3-Feb-2009	No		**5**	**4**	
26-Jan-2009	33451	28-Jan-2009	2008-4924*, 2009-0298	29-Jan-2009	15267 <-> WEB-ACTIVEX MW6 Technologies Barcode ActiveX clsid unicode access (web-activex.rules, High)	3-Feb-2009	No		**5**	**4**	
26-Jan-2009	33451	28-Jan-2009	2008-4924*, 2009-0298	29-Jan-2009	15268 <-> WEB-ACTIVEX MW6 Technologies Barcode ActiveX function call access (web-activex.rules, High)	3-Feb-2009	No		**5**	**4**	
26-Jan-2009	33451	28-Jan-2009	2008-4924*, 2009-0298	29-Jan-2009	15269 <-> WEB-ACTIVEX MW6 Technologies Barcode ActiveX function call unicode access (web-activex.rules, High)	3-Feb-2009	No		**5**	**4**	
29-Oct-2008			2008-4926*	6-Nov-2008	15270 <-> WEB-ACTIVEX MW6 Technologies PDF417 ActiveX clsid access (web-activex.rules, High)	3-Feb-2009	No			**87**	
29-Oct-2008			2008-4926*	6-Nov-2008	15271 <-> WEB-ACTIVEX MW6 Technologies PDF417 ActiveX clsid unicode access (web-activex.rules, High)	3-Feb-2009	No			**87**	
29-Oct-2008			2008-4926*	6-Nov-2008	15272 <-> WEB-ACTIVEX MW6 Technologies PDF417 ActiveX function call access (web-activex.rules, High)	3-Feb-2009	No			**87**	
29-Oct-2008			2008-4926*	6-Nov-2008	15273 <-> WEB-ACTIVEX MW6 Technologies PDF417 ActiveX function call unicode access (web-activex.rules, High)	3-Feb-2009	No			**87**	
29-Oct-2008			2008-4925	6-Nov-2008	15274 <-> WEB-ACTIVEX MW6 Technologies DataMatrix ActiveX clsid access (web-activex.rules, High)	3-Feb-2009	No			**87**	
29-Oct-2008			2008-4925	6-Nov-2008	15275 <-> WEB-ACTIVEX MW6 Technologies DataMatrix ActiveX clsid unicode access (web-activex.rules, High)	3-Feb-2009	No			**87**	
29-Oct-2008			2008-4925	6-Nov-2008	15276 <-> WEB-ACTIVEX MW6 Technologies DataMatrix ActiveX function call access (web-activex.rules, High)	3-Feb-2009	No			**87**	
29-Oct-2008			2008-4925	6-Nov-2008	15277 <-> WEB-ACTIVEX MW6 Technologies DataMatrix ActiveX function call unicode access (web-activex.rules, High)	3-Feb-2009	No			**87**	
29-Oct-2008			2008-4923	6-Nov-2008	15278 <-> WEB-ACTIVEX MW6 Technologies Aztec ActiveX clsid access (web-activex.rules, High)	3-Feb-2009	No			**87**	
29-Oct-2008			2008-4923	6-Nov-2008	15279 <-> WEB-ACTIVEX MW6 Technologies Aztec ActiveX clsid unicode access (web-activex.rules, High)	3-Feb-2009	No			**87**	
29-Oct-2008			2008-4923	6-Nov-2008	15280 <-> WEB-ACTIVEX MW6 Technologies Aztec ActiveX function call access (web-activex.rules, High)	3-Feb-2009	No			**87**	
29-Oct-2008			2008-4923	6-Nov-2008	15281 <-> WEB-ACTIVEX MW6 Technologies Aztec ActiveX function call unicode access (web-activex.rules, High)	3-Feb-2009	No			**87**	
26-Jan-2009	33453	28-Jan-2009	2009-0301	28-Jan-2009	15282 <-> WEB-ACTIVEX FlexCell Grid ActiveX clsid access (web-activex.rules, High)	3-Feb-2009	No		**5**	**5**	
26-Jan-2009	33453	28-Jan-2009	2009-0301	28-Jan-2009	15283 <-> WEB-ACTIVEX FlexCell Grid ActiveX clsid unicode access (web-activex.rules, High)	3-Feb-2009	No		**5**	**5**	
27-May-2008			2008-0958***	29-May-2008	15284 <-> WEB-ACTIVEX NCTAudioGrabber2 ActiveX clsid access (web-activex.rules, High)	3-Feb-2009	No			**0**	
27-May-2008			2008-0958***	29-May-2008	15285 <-> WEB-ACTIVEX NCTAudioGrabber2 ActiveX clsid unicode access (web-activex.rules, High)	3-Feb-2009	No			**0**	
27-May-2008			2008-0958***	29-May-2008	15286 <-> WEB-ACTIVEX NCTAudioGrabber2 ActiveX function call access (web-activex.rules, High)	3-Feb-2009	No			**0**	

27-May-2008			2009-0958***	29-May-2008	15287 <-> WEB-ACTIVEX NCTAudioGrabber2 ActiveX function call unicode access (web-activex.rules, High)	3-Feb-2009	No			0	
27-May-2008			2010-0958***	29-May-2008	15288 <-> WEB-ACTIVEX NCTAudioInformation2 ActiveX clsid access (web-activex. rules, High)	3-Feb-2009	No			0	
27-May-2008			2011-0958***	29-May-2008	15289 <-> WEB-ACTIVEX NCTAudioInformation2 ActiveX clsid unicode access (web-activex.rules, High)	3-Feb-2009	No			0	
27-May-2008			2012-0958***	29-May-2008	15290 <-> WEB-ACTIVEX NCTAudioInformation2 ActiveX function call access (web-activex.rules, High)	3-Feb-2009	No			0	
27-May-2008			2013-0958***	29-May-2008	15291 <-> WEB-ACTIVEX NCTAudioInformation2 ActiveX function call unicode access (web-activex.rules, High)	3-Feb-2009	No			0	
					15292 <-> CHAT QQ protocol detected - version 2006 (chat. rules, High)	3-Feb-2009					
					15293 <-> CHAT QQ protocol detected - version 2008 (chat. rules, High)	3-Feb-2009					
					15295 <-> SPECIFIC-THREATS Trojan.Bankpatch.C configuration attempt (specific-threats.rules, High)	3-Feb-2009					
					15296 <-> SPECIFIC-THREATS Trojan.Bankpatch.C malicious file download attempt (specific-threats.rules, High)	3-Feb-2009					
					15297 <-> SPECIFIC-THREATS Trojan.Bankpatch.C report home attempt (specific-threats.rules, High)	3-Feb-2009					
	33233	17-Jan-2009			15228 <-> WEB-ACTIVEX Ciansoft PDFBuilderX ActiveX clsid access (web-activex. rules, High)	27-Jan-2009	No		10		
	33233	17-Jan-2009			15229 <-> WEB-ACTIVEX Ciansoft PDFBuilderX ActiveX clsid unicode access (web-activex.rules, High)	27-Jan-2009	No		10		
	23811, 33238, 33243, 33245	27-Jan-2009			15230 <-> WEB-ACTIVEX Office Viewer 2 ActiveX clsid access (web-activex.rules, High)	27-Jan-2009	No		0		
	23811, 33238, 33243, 33245	27-Jan-2009			15231 <-> WEB-ACTIVEX Office Viewer 2 ActiveX clsid unicode access (web-activex. rules, High)	27-Jan-2009	No		0		
14-Jan-2009	33272	16-Jan-2009	2009-0134	29-Jan-2009	15232 <-> WEB-ACTIVEX Easy Grid ActiveX clsid access (web-activex.rules, High)	27-Jan-2009	No		11	-2	
14-Jan-2009	33272	16-Jan-2009	2009-0134	29-Jan-2009	15233 <-> WEB-ACTIVEX Easy Grid ActiveX clsid unicode access (web-activex. rules, High)	27-Jan-2009	No		11	-2	
14-Jan-2009	33272	16-Jan-2009	2009-0134	29-Jan-2009	15234 <-> WEB-ACTIVEX Easy Grid ActiveX function call access (web-activex.rules, High)	27-Jan-2009	No		11	-2	
14-Jan-2009	33272	16-Jan-2009	2009-0134	29-Jan-2009	15235 <-> WEB-ACTIVEX Easy Grid ActiveX function call unicode access (web-activex. rules, High)	27-Jan-2009	No		11	-2	
23-Jan-2009	33408*	19-Jan-2009	2008-5260*	16-Jan-2009	15243 <-> WEB-ACTIVEX AXIS Camera ActiveX clsid access (web-activex.rules, High)	27-Jan-2009	AxisCamControl ActiveX Control Buffer Overflow Vulnerability 35454	24-Jan-2009	8	11	3
23-Jan-2009	33408*	19-Jan-2009	2008-5260*	16-Jan-2009	15244 <-> WEB-ACTIVEX AXIS Camera ActiveX clsid unicode access (web-activex. rules, High)	27-Jan-2009	AxisCamControl ActiveX Control Buffer Overflow Vulnerability 35,454	24-Jan-2009	8	11	3
23-Jan-2009	33408*	19-Jan-2009	2008-5260*	16-Jan-2009	15245 <-> WEB-ACTIVEX AXIS Camera ActiveX function call access (web-activex.rules, High)	27-Jan-2009	AxisCamControl ActiveX Control Buffer Overflow Vulnerability 35,454	24-Jan-2009	8	11	3
23-Jan-2009	33408*	19-Jan-2009	2008-5260*	16-Jan-2009	15246 <-> WEB-ACTIVEX AXIS Camera ActiveX function call unicode access (web-activex.rules, High)	27-Jan-2009	AxisCamControl ActiveX Control Buffer Overflow Vulnerability 35,454	24-Jan-2009	8	11	3
	33345	20-Jan-2009			15247 <-> WEB-ACTIVEX JamDTA ActiveX clsid access (web-activex.rules, High)	27-Jan-2009	No		7		
	33345	20-Jan-2009			15248 <-> WEB-ACTIVEX JamDTA ActiveX clsid unicode access (web-activex.rules, High)	27-Jan-2009	No		7		
	33348, 33349	20-Jan-2009			15249 <-> WEB-ACTIVEX SmartVMD ActiveX clsid access (web-activex.rules, High)	27-Jan-2009	No		7		

	33348, 33349	20-Jan-2009			15250 <-> WEB-ACTIVEX SmartVMD ActiveX clsid unicode access (web-activex. rules, High)	27-Jan-2009	No		7		
	33318	19-Jan-2009			15251 <-> WEB-ACTIVEX MetaProducts MetaTreeX ActiveX clsid access (web-activex.rules, High)	27-Jan-2009	No		8		
	33318	19-Jan-2009			15252 <-> WEB-ACTIVEX MetaProducts MetaTreeX ActiveX clsid unicode access (web-activex.rules, High)	27-Jan-2009	No		8		
	33318	19-Jan-2009			15253 <-> WEB-ACTIVEX MetaProducts MetaTreeX ActiveX function call access (web-activex.rules, High)	27-Jan-2009	No		8		
	33318	19-Jan-2009			15254 <-> WEB-ACTIVEX MetaProducts MetaTreeX ActiveX function call unicode access (web-activex.rules, High)	27-Jan-2009	No		8		
14-Jan-2009	33177*	8-Jan-2009	2008-5444*	14-Jan-2009	15255 <-> ORACLE Secure Backup msgid 0x901 username field overflow attempt (oracle.rules, High)	27-Jan-2009	Oracle WebLogic Server Plug-in Remote Overflow (1166189) 35374	15-Jan-2009	19	13	12
14-Jan-2009	33177*	8-Jan-2009	2008-4014*	14-Jan-2009	15256 <-> ORACLE BPEL process manager XSS injection attempt (oracle.rules, High)	27-Jan-2009	No		19	13	
14-Jan-2009	33177*	8-Jan-2009	2008-4006*	14-Jan-2009	15257 <-> ORACLE Secure Backup common.php variable based command injection attempt (oracle.rules, High)	27-Jan-2009	Oracle Secure Backup Administration Server login.php Command Injection Vulnerability 35363		19	13	
14-Jan-2009	33177*	8-Jan-2009	2008-5449*	14-Jan-2009	15258 <-> ORACLE Secure Backup login.php variable based command injection attempt (oracle.rules, High)	27-Jan-2009	No		19	13	
14-Jan-2009	33177*	8-Jan-2009	2008-5448*	14-Jan-2009	15261 <-> ORACLE Secure Backup exec_qr command injection attempt (oracle.rules, High)	27-Jan-2009	Oracle Secure Backup Administration Server login.php Command Injection Vulnerability 35363		19	13	
14-Jan-2009	33177*	8-Jan-2009	2008-5448*	14-Jan-2009	15262 <-> ORACLE Secure Backup POST exec_qr command injection attempt (oracle.rules, High)	27-Jan-2009	Oracle Secure Backup Administration Server login.php Command Injection Vulnerability 35363		19	13	
14-Jan-2009	33177*	8-Jan-2009	2008-5440*	14-Jan-2009	15264 <-> WEB-CGI Oracle TimesTen In-Memory Database evtdump CGI module format string exploit attempt (web-cgi.rules, High)	27-Jan-2009	No		19	13	
	33233	17-Jan-2009			15228 <-> WEB-ACTIVEX Ciansoft PDFBuilderX ActiveX clsid access (web-activex. rules, High)	20-Jan-2009	No		3		
	33233	17-Jan-2009			15229 <-> WEB-ACTIVEX Ciansoft PDFBuilderX ActiveX clsid unicode access (web-activex.rules, High)	20-Jan-2009	No		3		
	23811, 33238, 33243, 33245*	13-Jan-2009			15230 <-> WEB-ACTIVEX Office Viewer 2 ActiveX clsid access (web-activex.rules, High)	20-Jan-2009	No		7		
	23811, 33238, 33243, 33245*	13-Jan-2009			15231 <-> WEB-ACTIVEX Office Viewer 2 ActiveX clsid unicode access (web-activex. rules, High)	20-Jan-2009	No		7		
14-Jan-2009	33272	16-Jan-2009	2009-0134*	16-Jan-2009	15232 <-> WEB-ACTIVEX Easy Grid ActiveX clsid access (web-activex.rules, High)	20-Jan-2009	No		4	4	
14-Jan-2009	33272	16-Jan-2009	2009-0134*	16-Jan-2009	15233 <-> WEB-ACTIVEX Easy Grid ActiveX clsid unicode access (web-activex. rules, High)	20-Jan-2009	No		4	4	
14-Jan-2009	33272	16-Jan-2009	2009-0134*	16-Jan-2009	15234 <-> WEB-ACTIVEX Easy Grid ActiveX function call access (web-activex.rules, High)	20-Jan-2009	No		4	4	
14-Jan-2009	33272	16-Jan-2009	2009-0134*	16-Jan-2009	15235 <-> WEB-ACTIVEX Easy Grid ActiveX function call unicode access (web-activex. rules, High)	20-Jan-2009	No		4	4	
22-Apr-2007	23620**	27-Feb-2008	2007-2193	13-Nov-2008	15236 <-> WEB-CLIENT ACD Systems ACDSee XPM file format overflow attempt (web-client.rules, High)	20-Jan-2009	No		0	67	
24-Apr-2007	23608**	24-Oct-2007	2007-2175	5-Sep-2008	15238 <-> SPECIFIC-THREATS Apple QuickTime for Java toQTPointer function memory corruption attempt (specific-threats.rules, High)	20-Jan-2009	Quicktime < 7.1.6 (Mac OS X) 25122 QuickTime < 7.1.6 (Windows) 25123**	30-May-2007	0	135	0
30-Nov-2008	32545	31-Dec-2008	2008-5276	29-Jan-2009	15241 <-> MULTIMEDIA VideoLAN VLC real.c ReadRealIndex real demuxer integer overflow attempt (multimedia.rules, High)	20-Jan-2009	VLC Media Player 0.9.x < 0.9.8a RealMedia Processing Remote Integer Overflow 35068	9-Dec-2008	20	-9	41

7-Jan-2009	33147*	7-Jan-2009	2008-0067	31-Jan-2009	15242 <-> WEB-CLIENT HP OpenView Network Node Manager Toolbar.exe HTTP request buffer overflow attempt (web-client.rules, High)	20-Jan-2009	No		**13**	**-10**	
					15227 <-> NETBIOS-DG SMB Trans2 OPEN2 unicode andx param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15226 <-> NETBIOS-DG SMB Trans2 OPEN2 andx param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15225 <-> NETBIOS SMB Trans2 OPEN2 andx param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15224 <-> NETBIOS SMB Trans2 OPEN2 unicode andx param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15223 <-> NETBIOS-DG SMB Trans2 OPEN2 unicode param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15222 <-> NETBIOS-DG SMB Trans2 OPEN2 param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15221 <-> NETBIOS SMB Trans2 OPEN2 param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15220 <-> NETBIOS SMB Trans2 OPEN2 unicode param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15219 <-> NETBIOS-DG SMB Trans2 OPEN2 unicode andx max_param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15218 <-> NETBIOS SMB Trans2 OPEN2 andx max_param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15217 <-> NETBIOS SMB Trans2 OPEN2 unicode andx max_param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15216 <-> NETBIOS-DG SMB Trans2 OPEN2 andx max_param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15215 <-> NETBIOS-DG SMB Trans2 OPEN2 unicode max_param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15214 <-> NETBIOS SMB Trans2 OPEN2 max_param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15213 <-> NETBIOS SMB Trans2 OPEN2 unicode max_param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15212 <-> NETBIOS-DG SMB Trans2 OPEN2 max_param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15211 <-> NETBIOS-DG SMB NT Trans NT CREATE andx max_param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15210 <-> NETBIOS SMB NT Trans NT CREATE andx max_param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15209 <-> NETBIOS SMB NT Trans NT CREATE unicode andx max_param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15208 <-> NETBIOS-DG SMB NT Trans NT CREATE unicode andx max_param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15207 <-> NETBIOS-DG SMB NT Trans NT CREATE max_param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15206 <-> NETBIOS SMB NT Trans NT CREATE max_param_count underflow attempt (netbios.rules)	13-Jan-2009					

					15205 <-> NETBIOS SMB NT Trans NT CREATE unicode max_param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15204 <-> NETBIOS-DG SMB NT Trans NT CREATE unicode max_param_count underflow attempt (netbios. rules)	13-Jan-2009					
					15203 <-> NETBIOS SMB NT Trans NT CREATE andx param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15202 <-> NETBIOS-DG SMB NT Trans NT CREATE unicode andx param_count underflow attempt (netbios. rules)	13-Jan-2009					
					15201 <-> NETBIOS-DG SMB NT Trans NT CREATE andx param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15200 <-> NETBIOS SMB NT Trans NT CREATE unicode andx param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15199 <-> NETBIOS SMB NT Trans NT CREATE param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15198 <-> NETBIOS-DG SMB NT Trans NT CREATE unicode param_count underflow attempt (netbios. rules)	13-Jan-2009					
					15197 <-> NETBIOS-DG SMB NT Trans NT CREATE param_count underflow attempt (netbios.rules)	13-Jan-2009					
					15196 <-> NETBIOS SMB NT Trans NT CREATE unicode param_count underflow attempt (netbios.rules	13-Jan-2009					
	32814	18-Dec-2008			15159 <-> WEB-ACTIVEX Evans FTP ActiveX clsid access (web-activex.rules, High)	6-Jan-2009	No		**18**		
	32814	18-Dec-2008			15160 <-> WEB-ACTIVEX Evans FTP ActiveX clsid unicode access (web-activex. rules, High)	6-Jan-2009	No		**18**		
	32814	18-Dec-2008			15161 <-> WEB-ACTIVEX Evans FTP ActiveX function call access (web-activex.rules, High)	6-Jan-2009	No		**18**		
	32814	18-Dec-2008			15162 <-> WEB-ACTIVEX Evans FTP ActiveX function call unicode access (web-activex.rules, High)	6-Jan-2009	No		**18**		
					15167 <-> POLICY Suspicious .cn dns query (policy.rules, High)	6-Jan-2009					
					15168 <-> POLICY Suspicious .ru dns query (policy.rules, High)	6-Jan-2009					
					15169 <-> POLICY XBOX Live Kerberos Authenthication Request (policy.rules, High)	6-Jan-2009					
					15170 <-> POLICY XBOX Netflix client active (policy. rules, High)	6-Jan-2009					
					15171 <-> POLICY XBOX Marketplace http request (policy.rules, High)	6-Jan-2009					
					15172 <-> POLICY XBOX avatar retrieval request (policy. rules, High)	6-Jan-2009					
17-Dec-2008	32901	17-Dec-2008	2008-5691*	19-Dec-2008	15173 <-> WEB-ACTIVEX Phoenician Casino ActiveX clsid access (web-activex. rules, High)	6-Jan-2009	No		**19**	**17**	
17-Dec-2008	32901	17-Dec-2008	2008-5691*	19-Dec-2008	15174 <-> WEB-ACTIVEX Phoenician Casino ActiveX clsid unicode access (web-activex.rules, High)	6-Jan-2009	No		**19**	**17**	
17-Dec-2008	32901	17-Dec-2008	2008-5691*	19-Dec-2008	15175 <-> WEB-ACTIVEX Phoenician Casino ActiveX function call access (web-activex.rules, High)	6-Jan-2009	No		**19**	**17**	
17-Dec-2008	32901	17-Dec-2008	2008-5691*	19-Dec-2008	15176 <-> WEB-ACTIVEX Phoenician Casino ActiveX function call unicode access (web-activex.rules, High)	6-Jan-2009	No		**19**	**17**	
18-Dec-2008	32950, 32965	29-Dec-2008	2008-2434, 2008-2435	6-Jan-2009	15177 <-> WEB-ACTIVEX Trend Micro HouseCall ActiveX clsid access (web-activex. rules, High)	6-Jan-2009	No		**7**	**0**	

18-Dec-2008	32950, 32965	29-Dec-2008	2008-2434, 2008-2435	6-Jan-2009	15178 <-> WEB-ACTIVEX Trend Micro HouseCall ActiveX clsid unicode access (web-activex.rules, High)	6-Jan-2009	No		**7**	**0**	
18-Dec-2008	32950, 32965	29-Dec-2008	2008-2434, 2008-2435	6-Jan-2009	15179 <-> WEB-ACTIVEX Trend Micro HouseCall ActiveX function call access (web-activex.rules, High)	6-Jan-2009	No		**7**	**0**	
18-Dec-2008	32950, 32965	29-Dec-2008	2008-2434, 2008-2435	6-Jan-2009	15180 <-> WEB-ACTIVEX Trend Micro HouseCall ActiveX function call unicode access (web-activex.rules, High)	6-Jan-2009	No		**7**	**0**	
	33053	2-Jan-2009			15181 <-> WEB-ACTIVEX SaschArt SasCam Webcam Server ActiveX clsid access (web-activex.rules, High)	6-Jan-2009	No		**4**		
	33053	2-Jan-2009			15182 <-> WEB-ACTIVEX SaschArt SasCam Webcam Server ActiveX clsid unicode access (web-activex.rules, High)	6-Jan-2009	No		**4**		
					15183 <-> CHAT Yahoo messenger http link transmission attempt (chat. rules, High)	6-Jan-2009					
					15184 <-> CHAT MSN messenger http link transmission attempt (chat. rules, High)	6-Jan-2009					
					15185 <-> POLICY Nintendo Wii SSL Server Hello (policy. rules, High)	6-Jan-2009					
9-Oct-2008	31688*	9-Oct-2008	2008-3641*	10-Oct-2008	15186 <-> MISC Multiple vendors CUPS HPGL filter remote code execution attempt (misc.rules, High)	6-Jan-2009	CUPS < 1.3.9 Multiple Vulnerabilities 34385	11-Oct-2008	**87**	**86**	**85**
9-Oct-2008	31688*	9-Oct-2008	2008-3641*	10-Oct-2008	15187 <-> MISC Multiple vendors CUPS HPGL filter remote code execution attempt (misc.rules, High)	6-Jan-2009	CUPS < 1.3.9 Multiple Vulnerabilities 34385	11-Oct-2008	**87**	**86**	**85**
9-Oct-2008	31688*	9-Oct-2008	2008-3641*	10-Oct-2008	15188 <-> MISC Multiple vendors CUPS HPGL filter remote code execution attempt (misc.rules, High)	6-Jan-2009	CUPS < 1.3.9 Multiple Vulnerabilities 34385	11-Oct-2008	**87**	**86**	**85**
9-Oct-2008	31688*	9-Oct-2008	2008-3641*	10-Oct-2008	15189 <-> MISC Multiple vendors CUPS HPGL filter remote code execution attempt (misc.rules, High)	6-Jan-2009	CUPS < 1.3.9 Multiple Vulnerabilities 34385	11-Oct-2008	**87**	**86**	**85**
					15190 <-> WEB-MISC Youngzsoft CCProxy CONNECT Request buffer overflow attempt (web-misc. rules, High)	6-Jan-2009					
23-Sep-2008	31346*	23-Sep-2008	2008-4064*	24-Sep-2008	15191 <-> SPECIFIC-THREATS Mozilla Firefox animated PNG processing integer overflow (specific-threats.rules, High	6-Jan-2009	Firefox 3.x < 3.0.2 34267	24-Sep-2008	**103**	**102**	**102**

Bibliografía

[1] About the mitre corporation. URL http://www.mitre.org/about/index.html.

[2] Cve requirements and recommendations for cve compatibility. URL http://cve.mitre.org/compatible/questionnaires/96.html.

[3] Nessus: An automated network-based security scanner. URL http://www.sun.com/bigadmin/features/articles/nessus_scanner.html.

[4] The nessus attack scripting language reference guide. URL http://www.virtualblueness.net/nasl.html.

[5] Snort sourcefire homepage. URL http://www.snort.org/.

[6] Tenable network security. URL http://www.nessus.org/nessus/.

[7] Top 100 network security tools. URL http://sectools.org/.

[8] Bugtraq. 2009. URL http://en.wikipedia.org/wiki/Bugtraq.

[9] Richard J Barnett. Towards a taxonomy of network scanning techniques. 2008.

[10] Brian Caswell. *Snort Intrusion Detection and Prevention Toolkit.* Elsevier, 2007.

[11] Ron Gula. *Correlating IDS Alerts with Vulnerability Information.* Tenable Network Security, 2009.

[12] Metasploit LLC. *The Metasploit Framework User Guide*, 2008.

[13] Gordon "Fyodor" Lyon. *Nmap Network Scanning.* Insecure.Com LLC., 2009.

[14] Vijay Mukhi. *The Voyage To 0-Day*. http://www.vijaymukhi.com, 2009.

[15] osvdb. Open source vulnerability database. 2009. URL `http://www.osvdb.org`.

[16] Thomas Ptacek. Insertion, evasion and denial of service: Eluding network intrusion detection. 1998.

[17] CHOE SANG-HUN. Cyberattacks jam government and commercial web sites in u.s. and south korea. 2009. URL `http://www.nytimes.com/2009/07/09/technology/09cyber.html`.

[18] wikipedia. Exploit. 2009. URL `http://es.wikipedia.org/wiki/Exploit`.

[19] wikipedia. Openssh. 2009. URL `http://es.wikipedia.org/wiki/Openssh`.

[20] wikipedia. Postfix. 2009. URL `http://es.wikipedia.org/wiki/Postfix`.

[21] wikipedia. Samba (programa). 2009. URL `http://es.wikipedia.org/wiki/Samba`.

[22] wikipedia. Smtp. 2009. URL `http://es.wikipedia.org/wiki/SMTP`.

[23] wikipedia. Squid. 2009. URL `http://es.wikipedia.org/wiki/Squid`.

[24] Su-Yun Wu. Data mining-based intrusion detectors. *Expert Systems with Applications*, 2008.